# МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

***Факультет информационных технологий и робототехники (ФИТР)***

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 1** по дисциплине: ”Операционные системы и СП”

на тему: ”Установка, настройка и использование дистрибутива Linux”

|  |  |
| --- | --- |
| Выполнил**:** | студенты гр. 10701118 Воробей И.А. |
| Принял**:** | Станкевич С.Н. |

Минск 2019

**Цель лабораторной работы:** установить и настроить для комфортной работы один из дистрибутивов операционной системы Linux, изучить основные приёмы работы в данной операционной системе (ОС) и соответствующей графической оболочке (к примеру, в GNU/Linux или Unity), а также научиться пользоваться), а также научиться пользоваться встроенным в дистрибутив офисным приложением.

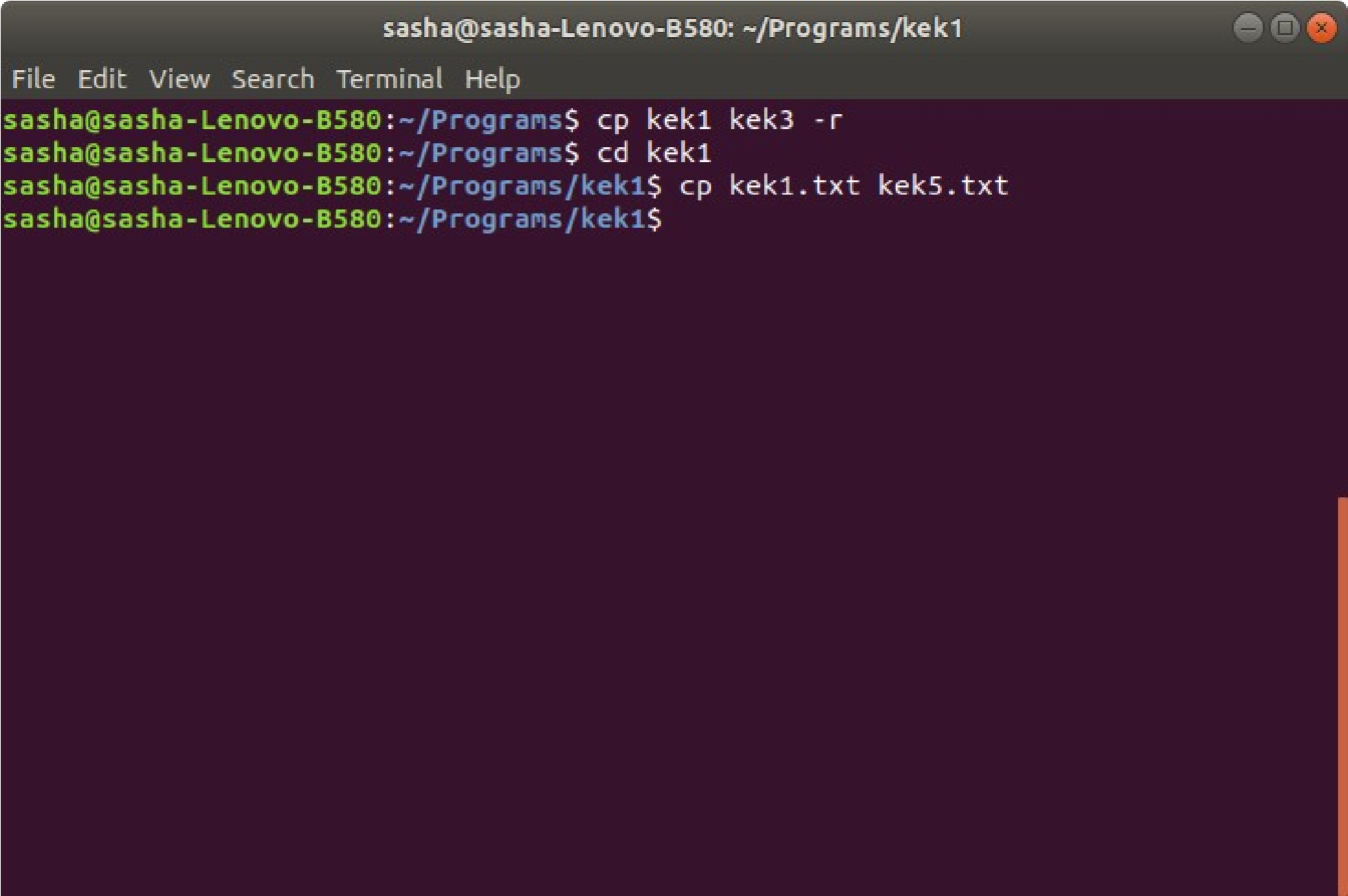
**Задание:**

1. Установить и настроить один из дистрибутивов ОС Linux (рекомендуется дистрибутив Ubuntu Linux). Ознакомиться с основными характеристиками дистрибутива и получить практические навыки работы в нём.
2. Изучить наиболее распространённые горячие клавиши (key), а также научиться пользоватьсяboard shortcuts) установленного дистрибутива.
3. Научиться использовать для своей работы встроенным офисным пакетом (кпримеру, Libre Office, Open Office, WPS Office или др.), а также с программами gedit, terminal и др.
4. Изучить основные команды для работы в терминале Linux (к примеру, ls, clear,cp, mv, cd, rm, cat, man и др.).

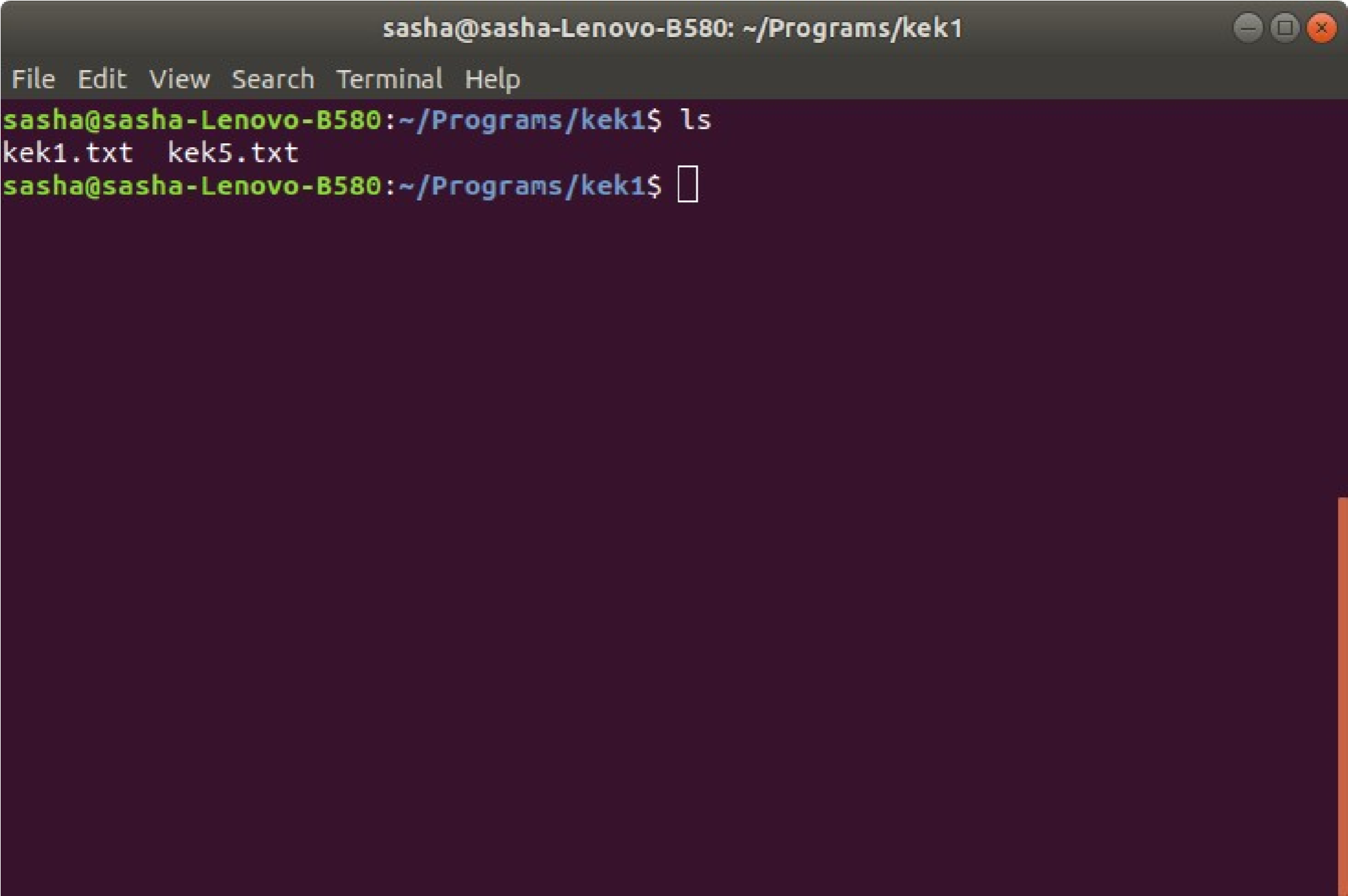
**Ход выполнения работы:**

1. Установили и настроили Ubuntu Linux
2. Изучили key), а также научиться пользоватьсяboard shortcuts
3. Научились пользоваться встроенным офисным пакетом, а также программамиgedit, terminal
4. Изучили основные команды для работы в терминале Linux

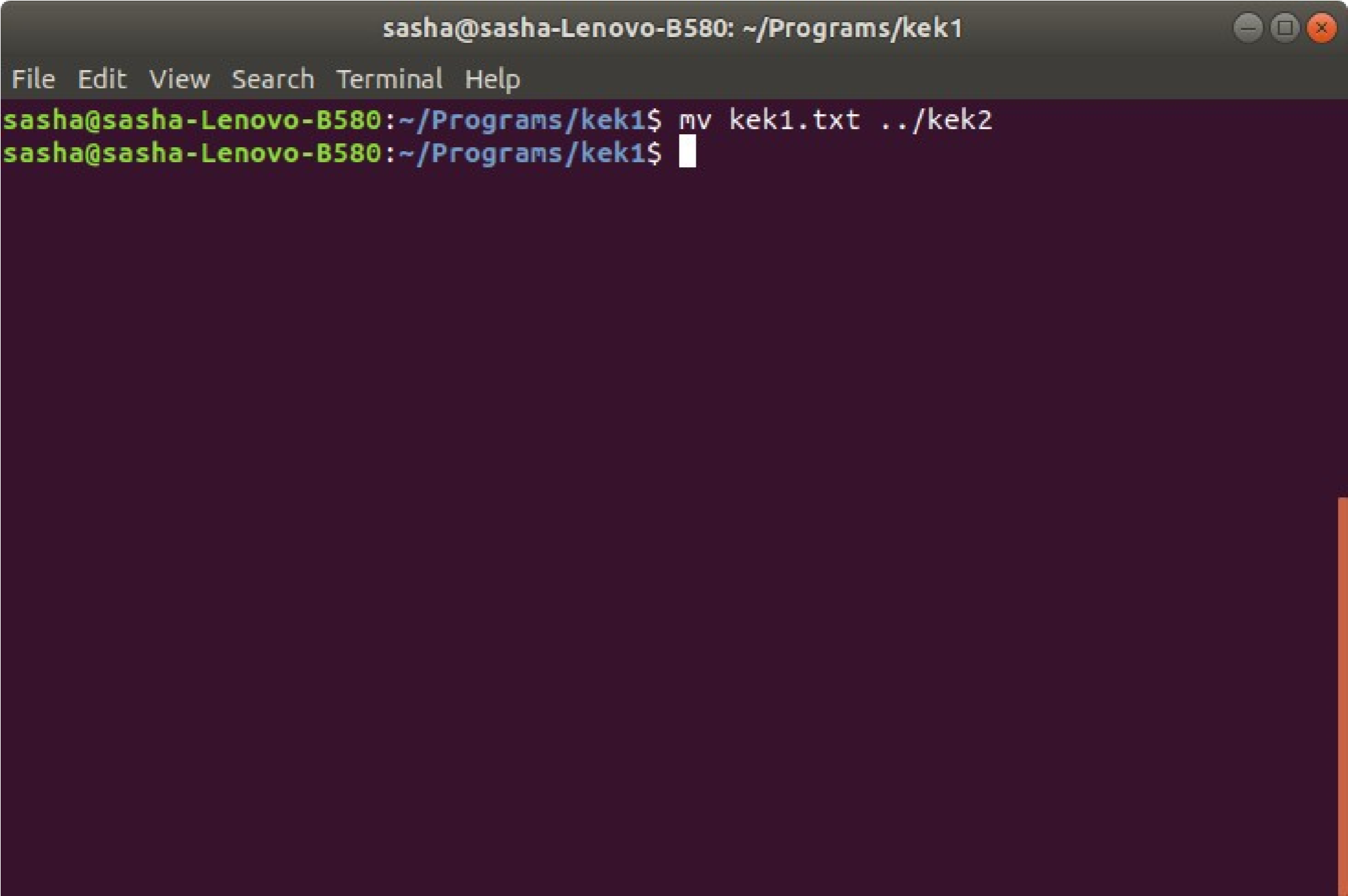
- cp



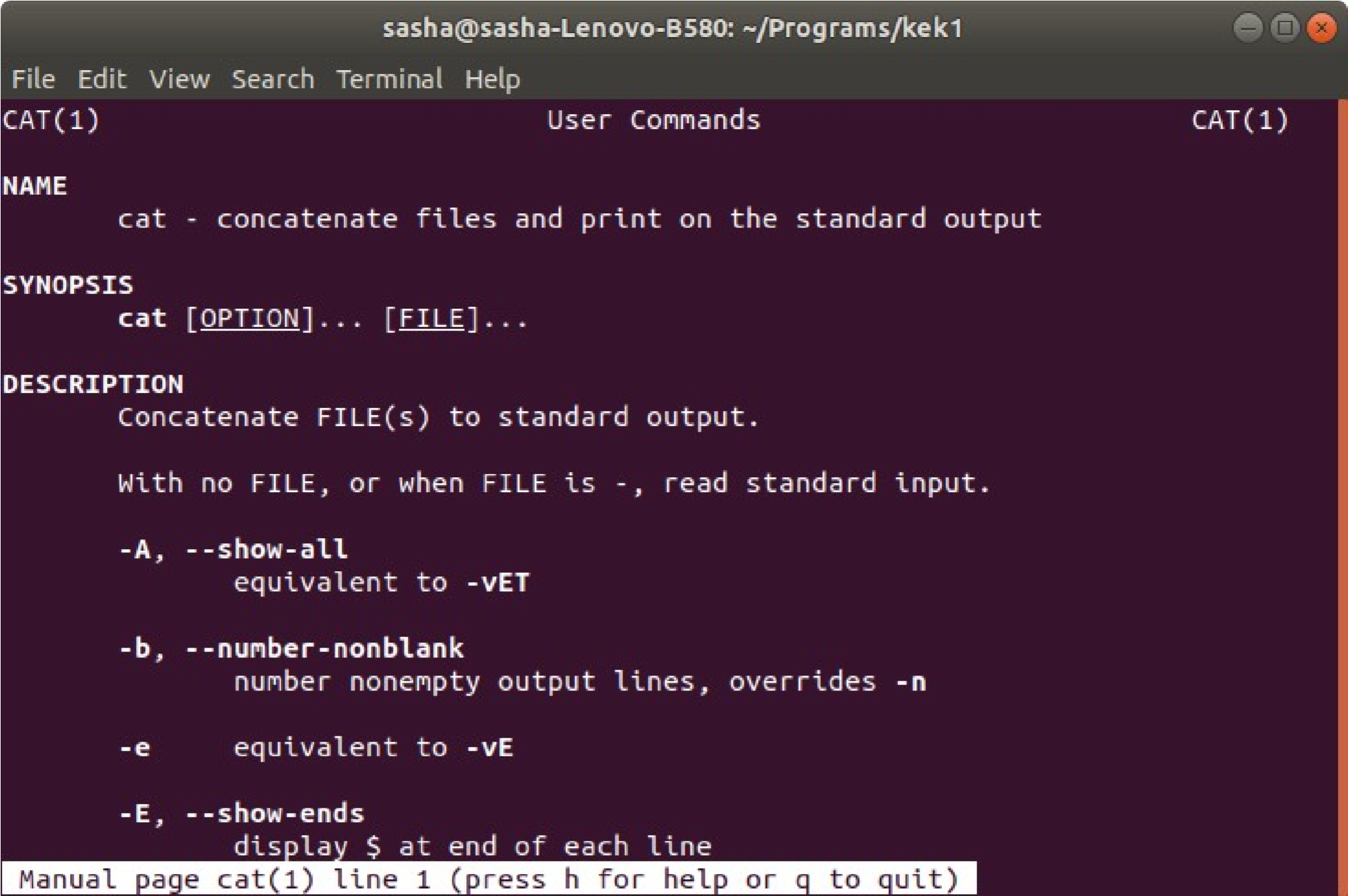
-ls



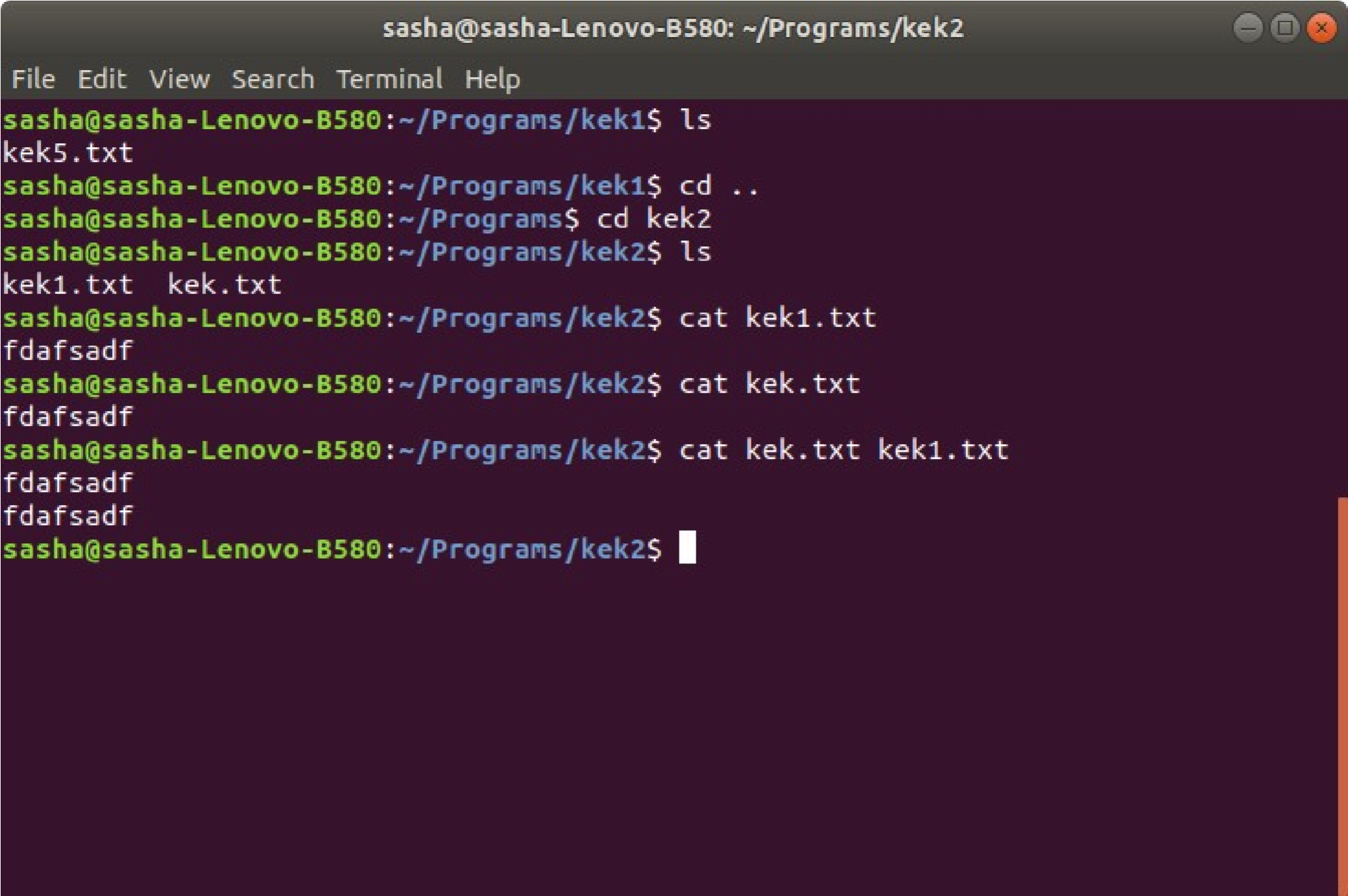
-mv



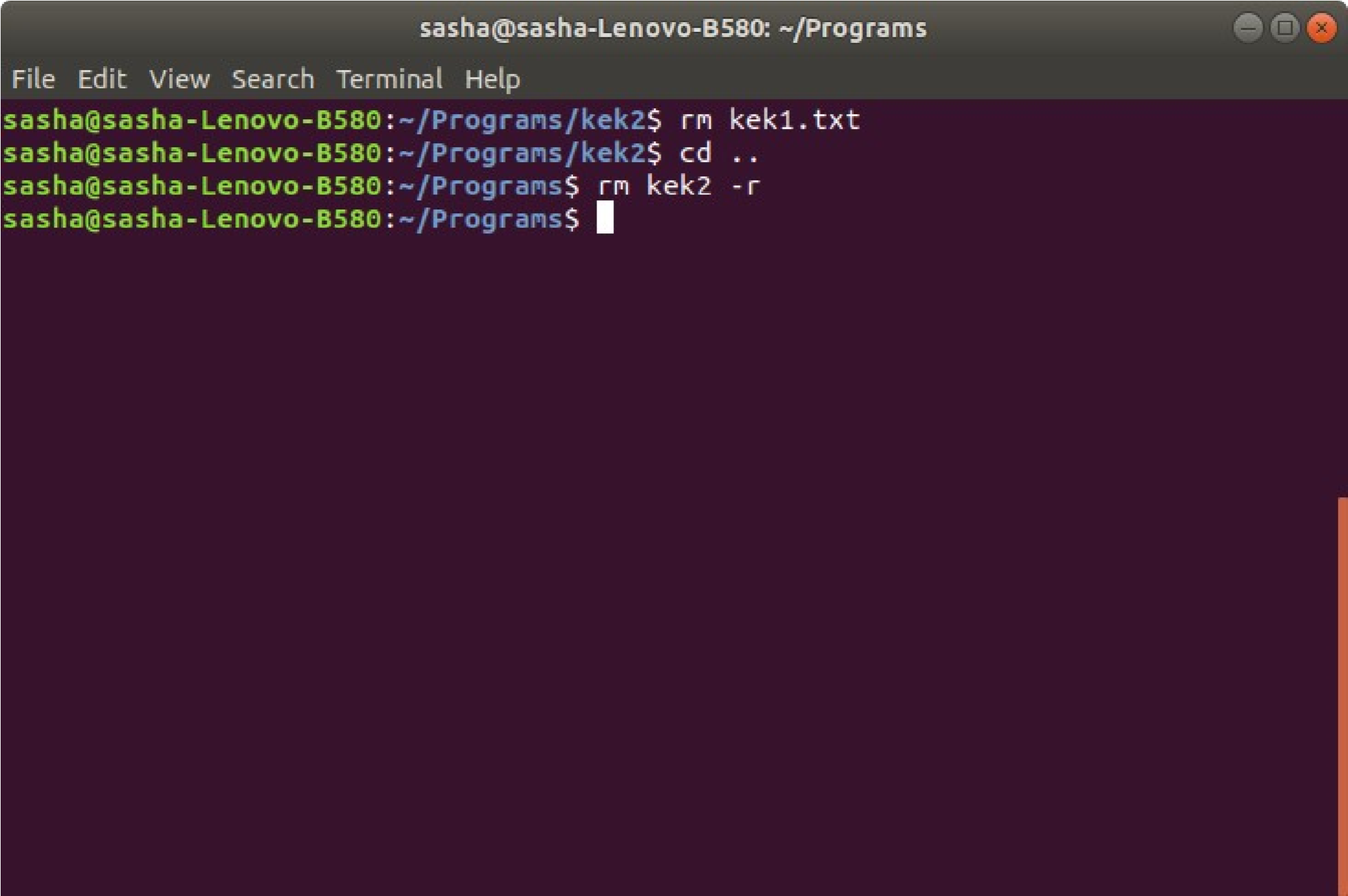
-man



-cat



-rm



**Контрольные вопросы:**

1. **Какие существуют типы ПО?**

* Системное
* Прикладное
* Underground

1. **Что такое ОС?**

ОС — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

1. **К какому типу ПО относится ОС?**

К системному ПО.

1. **Перечислите наиболее известные Вам разновидности ОС.**

* ОС семейства Windows
* ОС, основанные на ядре Linux
* Android
* IOS
* Blackberry), а также научиться пользоваться

1. **Перечислите разновидности настольных и мобильных ОС, а также их особенности.**

Linux и Windows — настольные ОС; Особенности настольных ОС:

* Интегрированная безопасность
* Поддержка многопроцессорных систем
* Повышенна защита от сбоев аппаратуры

Android, Bada(Samsung Electronics), Blackberry), а также научиться пользоваться, IOS, Windows Mobile — мобильные ОС.

Особенности мобильных ОС:

* Сочетают функциональность ОС для ПК и мобильных устройств: сенсорный экран, NFC, GPS

1. **Перечислите основные точки зрения определения ОС.**

* ОС — это набор программ, с помощью которых функционирует компьютер
* ОС — это интерфейс(мост) между «железом» и пользователем

1. **История развития ОС?**

Предшественники ОС: служебные программы(загрузчики и мониторы) — 1940г. 1950-1960 гг. формирование основных идей, определяющих функциональность ОС.

В 1965-1969 гг. разрабатывалась ОС Multics(MAC), дала начало ОС Unix.

В 1969г. разработана ОС UNICS(потом — UNIX) Брайаном Керниганом на ассемблере.

В 1974г. вышла первая коммерческая версия UNIX V6 на языке С.

В 1991г. официально вышла ОС Linux.

1. **Что представляет собой ОС семейства Unix (Linux)?**

Представляет собой операционную систему, в качестве ядра которой используется ядро Linux.

1. **Кто такой Линус Торвальдс?** Создатель ядра Linux.
2. **Почему в качестве логотипа ОС Linux был выбрал пингвин? Как его зовут?**

Создателю ядра Linux нравятся пингвины. Пингвина зовут TUX((T)orvalds (U)ni(X)).

1. **Что такое дистрибутив Linux и из каких частей он состоит?**

Состоит из:

* Ядро Linux
* Утилиты
* Библиотеки GNU
* Вспомогательные программы
* Документация
* Графическая подсистема
* Менеджер рабочего стола

1. **Перечислите популярные дистрибутивы Linux и их основное предназначение.**

* SteamOS — игровой дистрибутив(Valve)
* Ubuntu — получение полноценного и качественного рабочего стола на одном диске
* Kali — сетевой аудит и проверка на уязвимость
* Mint — удобный для использования новичками и пользовалями ПК

1. **Кто такой Ричард Столлман?**

Основатель движения свободного ПО, проекта GNU

1. **Что такое свободное ПО и сообщество Open Source?**Свободное ПО — бесплатное ПО, в общем доступе. Open source — с открытым исходным кодом.
2. **Что такое GPL-лицензия? Ее основные пункты?лицензия? Ее основные пункты?**

GPL — лицензия на свободное ПО, автор которого передает программу сообществу.

GPL предоставляет получателям программ:

* + Свободу запуска программы с любой целью
  + Свободу изучения того, как программа работает
  + Свободу распространения копий кода
  + Свободу улучшения программы

1. **Проект GNU и Фонд свободного ПО? Как расшифровывается аббревиатура GNU?**

Проект GNU — проект по разработке свободного ПО.

Фонд свободного ПО — организация для поддержи проекта GNU.

Были созданы Ричардом Столлманом. GNU — GNU Not Unix

1. **Чем занимается компания Canonical?**

Развивает и популяризирует проекты свободного ПО.

1. **Кто такой Марк Шаттлворт?**Основатель компании Canonical.
2. **Что означает слово «Ubuntu»? Что изображено на логотипе ОС Ubuntu Ubuntu»? Что изображено на логотипе ОС Ubuntu**

**Linux?**

Ubuntu — термин из нескольких южноафриканских языков, включая зулу и коса. На русский труднопереводимо, однако в общих чертах обозначает человечность по отношению к другим.

На логотипы изображены руки в кругу.

1. **Перечислите основные команды терминала Linux и их предназначение.**cp — копирование содержимого файла, с -r — копирование директории mv — переместить файл/директорию из одного места в другое cd — перемещение по директориям man — мануал по командам cat — выводит содержимое файлов rm — удаление clear — очистка консоли ls — показать содержимое директории

**Выводы:**

Установили и изучили дистрибутив Linux, основные программы и команды.

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 2

по дисциплине «***Операционные системы***»

тема: «Компиляция и отладка приложений в *Linux*»

|  |  |  |
| --- | --- | --- |
| Исполнители: |  | студенты группы 10701118  Воробей И.А. |
| Преподаватель: |  | Станкевич С.Н. |

Минск 2020

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**КОМПИЛЯЦИЯ И ОТЛАДКА ПРИЛОЖЕНИЙ В *LINUX***

**Цель работы**

Изучить инструментарий разработки приложений для Linuxи фундаментальные основы системного программирования с использованием утилит *gcc* (*g++*) и *gdb* для создания, компиляции и отладки приложений на языке программирования С/C++.

### Требования

1. Разработать приложение согласно варианту бригады. Во всех программах необходимо ввести N-строк, используя два способа ввода: через аргументы командной строки и стандартного ввода данных во время работы программы. Запустить программу в режиме отладки и продемонстрировать знания основных команд ***gdb***.
2. При разработки программы использовать модульный принцип.
3. Во время работы программы должны выводить на экран исходные и конечные данные.
4. При выполнении задания ***запрещается*** использовать интегрированные средства разработки (*Integrated Development Environment, IDE*). Рекомендуется задействовать любой текстовый редактор (к примеру, ***gedit***) и набор компиляторов GNU Compiler Collection (GCC), в частности, компилятор языка программирования С/C++ ***gcc/g++***.
5. При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention*)*.

**Индивидуальное задание:**

Разработать функцию для определения простого числа. Написать программу, которая тестирует работоспособность данной функции.

Ход выполнения лабораторной работы:

1. Установил соответствующий инструментарий для выполнения лабораторной работы (gcc, g++ и gdb).
2. Для удобства навигации по файлам установил дополнительно файловый менеджер GNOME Comander (gnome-commender).
3. Разработал алгоритм и блок-схему решения индивидуального задания.
4. В текстовом редакторе gedit набрал исходный код решения задания на языке программирования С++ и сохранил исходник с именем. Программа состоит из следующих файлов (см. рис. 1):
   * ***functions.h*** **–** файл заголовка, в котором описан прототип функции определения простого числа;
   * ***functions.cc*** **–** файл реализации функции определения простого числа;
   * ***test.cc*** **–** файл, в котором описана главная функция программы main**;**
5. Запустил терминал Linux и с помощью команды cd перешёл в папку местонахождения исходных кодов программы.
6. Для компиляции программы использовал компилятор g++ с вводом соответствующих параметров (см. рис. 2);
7. Для тестирования работоспособности запустил программу с соответствующими аргументами командной строки (см. рис. 2).
8. Для отладки программы использовал gdb (см. рис. 3).

Результаты выполнения лабораторной работы:

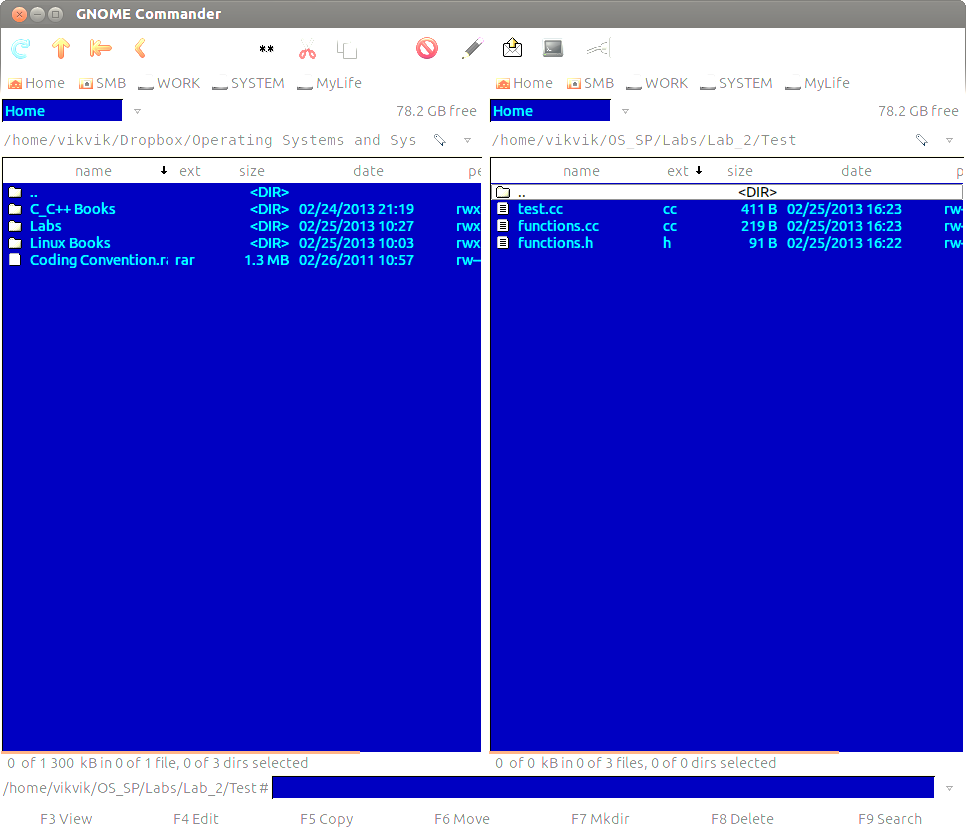


Рисунок 1 – Содержимое папки программы лабораторной работы № 2

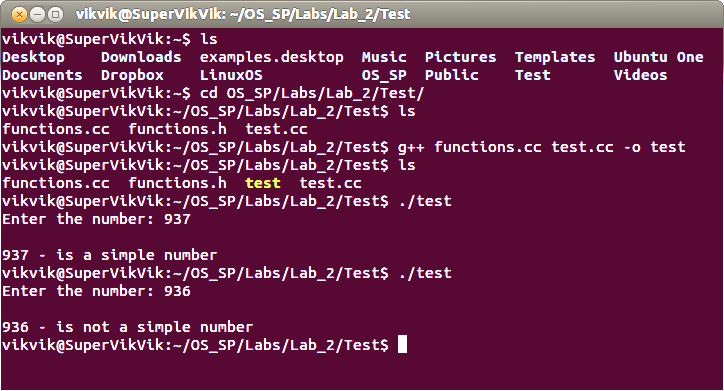


Рисунок 2 — Компиляция и выполнение программы test

Рисунок 3 — Запуск и отладка программы test с использованием gdb

**Контрольные вопросы:**

**1. Опишите базовый инструментарий системного программиста, проектирующий и разрабатывающий свои приложения на языке С/C++.**

* C ++ CMake инструменты для Linux
* Инструменты разработки встраиваемых и IoT
* Visual Studio
* IntelliCode
* IntelliTrace (только для предприятий)
* Подключение USB-устройства
* C ++ (v142) Универсальные инструменты для платформы Windows
* C ++ (v141) Универсальные инструменты для платформы Windows
* Графический отладчик и GPU-профилировщик для DirectX
* Windows 10 SDK (10.0.18362.0)
* Windows 10 SDK (10.0.17134.0)
* Windows 10 SDK (10.0.16299.0)
* Инструменты архитектуры и анализа

**2. Что такое исходный (*source*) код, а что такое исполняемый (*executable*, *binary*) код?**

Исходный код - текст компьютерной программы на каком-либо языке программирования или языке разметки, который может быть прочтён человеком. В обобщённом смысле — любые входные данные для транслятора. Исходный код транслируется в исполняемый код целиком до запуска программы при помощи компилятора или может исполняться сразу при помощи интерпретатора.

Исполняемый файл -  файл, содержащий программу в виде, в котором она может быть исполнена компьютером.

**3. Опишите в общем процесс разработки и исполнения программы под Linux? Что происходит за «сценой»?**

Только один процесс в системе рождается особенным способом — init — он порождается непосредственно ядром. Все остальные процессы появляются путём дублирования текущего процесса с помощью системного вызова fork(2). После выполнения fork(2) получаем два практически идентичных процесса за исключением следующих пунктов.

**4. Зачем нужен и какие действия выполняет препроцессор с исходным кодом программы?**

Препроцессор обрабатывает исходный код и передаёт на следующий шаг компиляции.

**5. Что такое компиляция (*compilation*)? С помощью чего и какой команды можно только скомпилировать код? Каким достоинством обладают компиляторы языка программирования С/C++?**

**Компиля́ция** — сборка программы, включающая трансляцию всех модулей программы, написанных на одном или нескольких исходных языках програмирования высокого уровня и/или языке ассемблера, в эквивалентные программные модули на низкоуровнем языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера) или непосредственно на машинном языке или ином двоичнокодов низкоуровневом командном языке и последующую сборку исполняемой машинной программы. Если компилятор генерирует исполняемую машинную программу на машинном языке, то такая программа непосредственно исполняется физической программируемой машиной (например компьютером).

bash$ gcc hello.c -o hello

Команда для компилирования кода.

Самое большое достоинства компилятора это его многоплатформенность.

**6. Что такое компоновка (линковка, сборка)?**

**Компоновщик (линкер)** связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы и сможем запустить в дальнейшем. Для того, чтобы понять как происходит связка, следует рассказать о таблице символов.

**7. Как запустить под семейство ОС Linux исполняемый файл на выполнение? От куда (с каких директорий) по умолчанию пытается запустить ОС Linux про-граммы?**

**8. Что такое *GCC*? Опишите наиболее востребования опции компилятора *gcc/g++*. Чем отличается *gcc* и *g++*?**

**GСС** - это свободно доступный оптимизирующий компилятор для языков C.

Опция **–с** – означает только компиляцию;

Опция **–о**  - задает имя исполняемого файла;

Опция **–Iname** - использовать при компоновке библиотеки libname.so

Опция **–g** - Поместить в объектный или исполняемый файл отладочную информацию для отладчика **gdb.**

**Опция –O – уровень оптимизации.**

**Опция –I -** Используется для добавления ваших собственных каталогов для поиска заголовочных файлов в процессе сборки.

Опция **–L** - Передается компоновщику. Используется для добавления ваших собственных каталогов для поиска библиотек в процессе сборки.

Опция -**l**  - Передается компоновщику. Используется для добавления ваших собственных библиотек для поиска в процессе сборки.

Компилятор **g++** распространяется по лицензии GNU, Фондом свободного программного обеспечения (FSF), для nix-подобных ОС и является C++ компилятором, который управляется с помощью командной строки. **g++** распространяется с nix системами, так что если вы работаете в ОС Unix или Linux, скорее всего в вашей системе уже установлен **g++**.

Опция –filename - После того, как исходный файл будет скомпилирован в исполняемый, на выходе мы получим файл с именем a и расширением .out — a.out.

Опция –o outputfile - перед запуском g++ можно указать в командной строке имя исполняемого файла, который получим на выходе. Для этого необходимо указать параметр -o и присвоить нужное имя файла.

Опция – wall-ansi- Чтобы g++ показывал предупреждения об ошибках, а также поддерживал существующие стандарты языка C++, рекомендую использовать флаги.

Опция –man g++ - если вы используете \*nix-подобные системы, вы также можете проверить другие опции компилятора g++, введя в командную строку следующую команду.

Отличия в том, что g++ - компилятор для C++, а компилятор gcc – компилятор С.

**9. Опишите основные команды отладчика *gdb* (выполнение программы, трассировка стека, просмотр переменных, вывод листинга программы, установка точек останова, вставка исправлений с помощью отладчика).**

* Компиляция программы
* Запуск отладчика
* Команды отладчика
* Доступ к исходному тексту
* Доступ к машинному коду
* Точки останова (breakpoints)
* Продолжение после останова
* Доступ к стеку вызова функций
* Доступ к переменной программы
* Доступ к памяти
* Доступ к регистрам
* Изменения в программе
* И другие команды.

**10. С помощью каких горячих клавиш в дистрибутиве ОС Ubuntu Linux можно вызвать наиболее востребованные команды и программы, к примеру, Linux-терминал?**

Клавиша **Super** – это клавиша с логотипом Windows

1. Ctrl + Alt + T – открыть терминал.
2. Super – открыть обзорный экран.
3. Super + L – блокировка экрана.
4. Super + D – свернуть окна и развернуть при повторном нажатии.
5. Super + A – открывает лаунчер для запуска приложений.
6. Super + M – просмотр уведомлений календаря.
7. Alt + Tab – переключение между открытыми приложениями.
8. Alt + Esc – переключение между приложениями текущего рабочего стола.
9. Super + ` - переключение между окнами одного приложения
10. Super +PgDown/PgUp – переключение между рабочими столами
11. Super + Shift + PgDown/PgUp – переместить окно на другой рабочий стол
12. Super + Space - сменить раскладку клавиатуры.
13. F2 – переименовать файл.
14. Alt + F2 – выполнить команду.
15. Ctrl + Alt + Del –выйти из системы.

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 3

по дисциплине «***Операционные системы***»

тема: **«Многофайловые проекты и шаблон MVC»**

|  |  |  |
| --- | --- | --- |
| Исполнители: |  | студенты группы 10701118  Воробей И.А.  Белявский Д.В. |
| Преподаватель: |  | Станкевич С.Н. |

Минск 2020

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Многофайловые проекты и шаблон MVC**

**Цель работы**

### Научиться грамотно проектировать и разрабатывать многофайловые проекты на языке программирования С/C++ с использованием архитектурного шаб-лона проектирования и разработки гибких и масштабируемых приложений Model-View-Controller (MVC).

### Требования

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования **MVC**.

2) Размерность матрицы в индивидуальном задании задаётся пользователем на стадии выполнения программы. Для повышения производительности про-граммы использовать динамическое выделение памяти.

3) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬ-НЫЕ переменные!

4) Каждое задание оформить в виде отдельной бизнес-функции.

5) Все функции должны быть сгруппированы по соответствующим отдельным файлам.

6) Все функции должны быть самодостаточные, т.е. при их разработке необхо-димо придерживаться принципа ***Single Responsibility Principle***.

**Индивидуальное задание:**

1)Поменять местами строку, содержащую элемент с наибольшим значением в матрице со строкой, содержащей элемент с наименьшим значением. Вывести на экран полученную матрицу. Для каждой строки с нулевым элементом на главной диагонали вывести ее номер и значение наибольшего из элементов этой строки.

2) Определить, является ли матрица симметричной (относительно главной диагонали). Вывести на экран соответствующее сообщение. Найти максимальный элемент среди стоящих на главной и побочной диагонали и поменять местами с элементом, стоящим на пересечении этих диагоналей.

3) Выведите номера столбцов, все элементы, которых четны. Для каждого столбца с отрицательным элементом на главной диагонали вывести его номер и сумму всех элементов этого столбца.

4) Среди строк заданной матрицы, содержащих только нечетные элементы, найти строку с максимальной суммой модулей элементов.

5) Среди столбцов заданной матрицы, содержащих только такие элементы, ко-торые по модулю не больше n, найти столбец с минимальным произведением элементов.

6) Выведите номера столбцов, элементы каждой из которых образуют монотон-ную последовательность (монотонно убывающую или монотонно возрастаю-щую).

7) Найти максимальный среди всех элементов тех строк заданной матрицы, ко-торые упорядочены (либо по возрастанию, либо по убыванию).

8) Найти номер строки заданной матрицы, в которой находится самая длинная серия (последовательность одинаковых элементов).

9) Найти строку заданной матрицы, в которой длина максимальной серии (по-следовательности одинаковых элементов) минимальна.

10) Найти максимальное из чисел, встречающихся в заданной матрице более од-ного раза.

11) Найти минимальное из чисел, встречающихся в заданной матрице ровно один раз.

12) Найти максимальное из чисел, встречающихся в заданной матрице ровно два раза.

13) Для заданной целочисленной матрицы найти максимум среди сумм элемен-тов диагоналей, параллельных главной диагонали матрицы.

14) Для заданной целочисленной матрицы найти минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

15) Подсчитать количество столбцов заданной матрицы, которые составлены из попарно различных чисел.

16) Подсчитать количество строк заданной матрицы, являющихся перестановкой чисел 1, 2, ..., 20.

17) Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Соседями элемента ajj в матрице назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j). Подсчитать количество локальных минимумов заданной матрицы.

18) Найти максимум среди всех локальных минимумов заданной матрицы. Эле-мент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Соседями элемента ajj в матрице назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j).

19) Найти минимум среди всех локальных максимумов заданной матрицы. Эле-мент матрицы называется локальным максимумом, если он строго больше всех имеющихся у него соседей. Соседями элемента ajj в матрице назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j).

20) Определить, становится ли симметричной (относительно главной диагонали) заданная матрица после замены на число 0 каждого локального минимума. Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Соседями элемента ajj в матрице назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j).

21) Определить, становится ли симметричной (относительно главной диагонали) заданная матрица после замены на число 0 каждого локального максимума. Элемент матрицы называется локальным максимумом, если он строго больше всех имеющихся у него соседей. Соседями элемента ajj в матрице назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j).

22) Две строки матрицы назовем похожими, если совпадают множества чисел, встречающихся в этих строках. Найти количество строк в максимальном мно-жестве попарно непохожих строк заданной матрицы.

23) Определить, является ли действительная матрица размера m x n ортонорми-рованной, т.е. такой, в которой скалярное произведение каждой пары раз-личных строк равно 0, а скалярное произведение каждой строки на себя равно 1. Вывести на экран соответствующее сообщение.

Ход выполнения лабораторной работы:

1. Установил соответствующий инструментарий для выполнения лабораторной работы (gcc, g++ и gdb).
2. Для удобства навигации по файлам установил дополнительно файловый менеджер GNOME Comander (gnome-commender).
3. Разработал алгоритм и блок-схему решения индивидуального задания.
4. В текстовом редакторе gedit набрал исходный код решения задания на языке программирования С++ и сохранил исходник с именем. Программа состоит из следующих файлов (см. рис. 1):
   * ***functions.h*** **–** файл заголовка, в котором описан прототип функции определения простого числа;
   * ***functions.cc*** **–** файл реализации функции определения простого числа;
   * ***test.cc*** **–** файл, в котором описана главная функция программы main**;**
5. Запустил терминал Linux и с помощью команды cd перешёл в папку местонахождения исходных кодов программы.
6. Для компиляции программы использовал компилятор g++ с вводом соответствующих параметров (см. рис. 2);
7. Для тестирования работоспособности запустил программу с соответствующими аргументами командной строки (см. рис. 2).
8. Для отладки программы использовал gdb (см. рис. 3).

Результаты выполнения лабораторной работы:

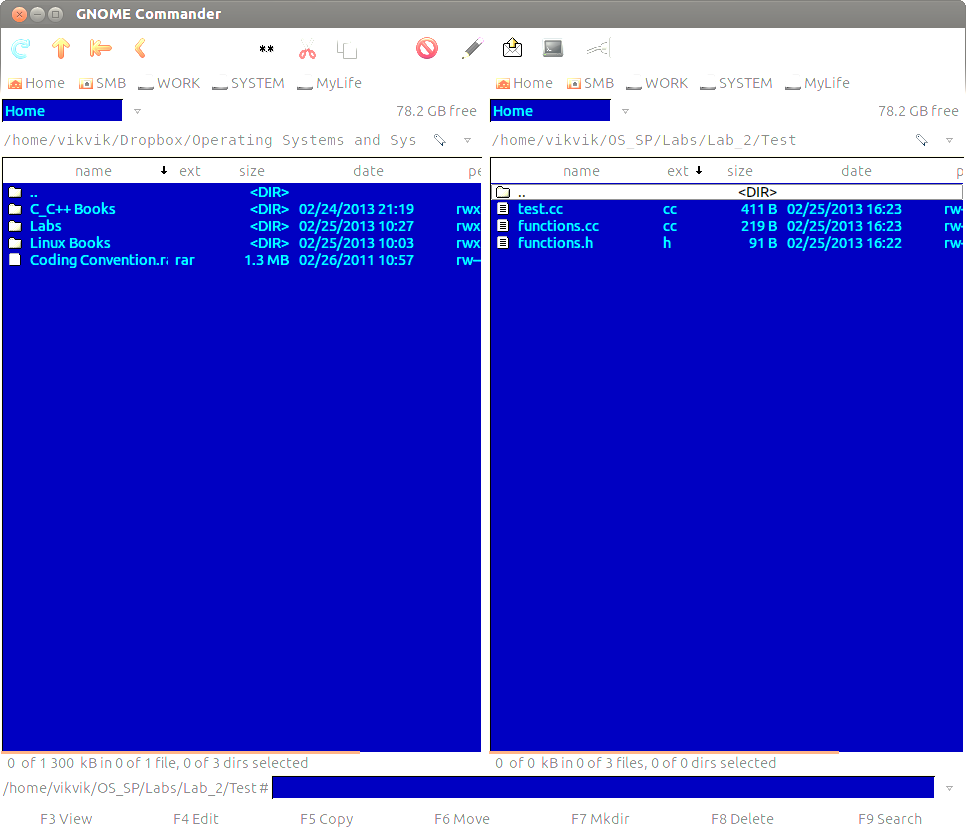


Рисунок 1 – Содержимое папки программы лабораторной работы № 2

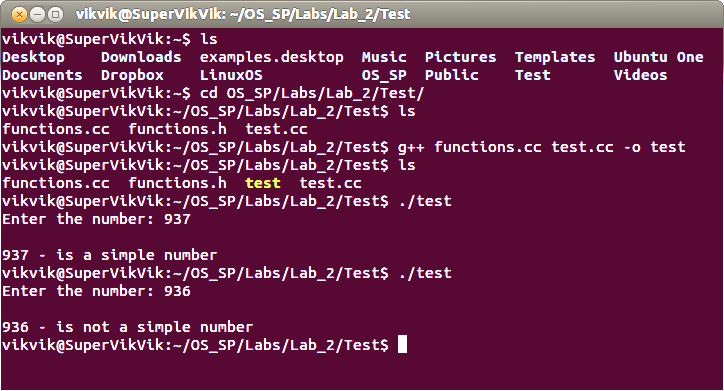


Рисунок 2 — Компиляция и выполнение программы test

Рисунок 3 — Запуск и отладка программы test с использованием gdb

**Контрольные вопросы:**

**1. Объясните основные термины (объявление (declaration), определение (definition), вызов (call)), которые связаны с описанием и использованием функций в языке C/C++, приведите пример. Зачем нужно определение функции?**

Термин  **declaration** означает декларирование идентификатора (обозначающего такой объект, как тип, переменная или функция) до того, как программист дал ему полное определение.

Пример:

void printThisInteger(int);

Директива #define определяет идентификатор и последовательность символов, которой будет за­мещаться данный идентификатор при его обнаружении в тексте программы.

Пример:

#define имя\_макроса последовательность\_символов.

#include <iostream> int main()

{ inline\_function(std::string s)

{ std::cout << "Hello, " << s << "!\n"; }

with\_name(make\_hello);

make\_hello("Vasiliy Pupkin!"); return 0; }

**2. Какие существенные преимущества имеют многофайловые проекты перед однофайловыми? Что такое сборка?**

Многофайловые сборки обладают тремя основными преимуществами перед однофайловыми:

1.В состав одной сборки можно включить тремя модули, код которых разработан на различных языках программирования. Более подробно об этом рассказано в подразделе «Модули» настоящей главы.

2. при запуске приложения через сеть очень удобно поместить код сборки и ресурсы в разные файлы. Таким образом, файлы будут подгружаться не все сразу, а по мере необходимости, при непосредственном обращении к ним, что существенно снизит трафик при первоначальной загрузке приложения. При повторном запуске приложения этот фактор не существенным, поскольку все данные уже будут записаны в кэш.

3.Также удобно выносить ресурсы за пределы сборки во время отладки и разработки приложения, поскольку для их редактирования не придется прибегать к специальным средствам, извлекающим их из сборок. Ресурсы будут доступны в виде в виде обычных дисковых файлов, которые можно будет редактировать привычным образом. При этом, чтобы увидеть внесенные изменения, приложение будет не нужно перекомпилировать.

**Сборка**  — процесс получения информационного продукта из исходного кода. Чаще всего включает компиляцию и компоновку, выполняется инструментами автоматизации.

**3. Почему в C/C++ раздельная компиляция? Опишите её преимущества и недостатки.**

У раздельной компиляции есть ряд преимуществ по сравнению с теми способами написания программ, в которых весь текст программы помещается в один огромный файл:

1) Такая организация программы поддерживает концепцию модульности, когда каждый из исходных файлов отвечает за один небольшой, законченный по смыслу фрагмент программы, что сильно упрощает и ускоряет разработку программ, а также резко упрощает командную разработку, когда над одной программой трудится (большой) коллектив разработчиков.

2) Если изменяется один исходный файл, перекомпилировать нужно только его. Правда, перекомпоновывать нужно все, но компоновка обычно выполняется быстрее компиляции.

3) Продавцы библиотек могут распространять свои библиотеки в скомпилированном виде, что затрудняет извлечение из них применяемых там алгоритмов и структур данных.

4) Наконец, иногда такой подход позволяет писать разные фрагменты программы на разных языках программирования. Здесь, правда, нужно соблюдать определенную осторожность. В качестве примера назовем два возможных препятствия: а) разные способы вызова функций, например, соглашения о том, в каком порядке параметры функции хранятся в памяти, и б) так называемое декорирование имен, т. е. преобразование имен функций на стадии компиляции, так что компоновщик видит не те имена функций, которые указаны в тексте программы, а результаты их преобразования компилятором/

**4. Опишите общий алгоритм процесса сборки многофайлового проекта и стадии его компиляции и выполнения.**

Стадии:

1. Препроцессинг

2.Компиляция

3.Сборка

**5. Что всегда будет присутствовать (сопровождать) в любой востребованной программе?**

Всегда будут главная функция, компиляция или интерпретация, сборка, рефакторинг.

**6. Что такое рефакторинг кода?**

**Рефакторинг кода** — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

**7. Опишите концепцию проектирования приложений с использованием шаблона Model-View-Controller. Зачем он применяется? Какие преимущества даёт использование данного шаблона при проектировании и реализации программного проекта?**

**Model-View-Controller** (**MVC**, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Основная цель применения этой концепции состоит в отделении бизнес-логики (*модели*) от её визуализации (*представления*, *вида*). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной *модели* можно присоединить несколько *видов*, при этом не затрагивая реализацию *модели*. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
2. Не затрагивая реализацию *видов*, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой *контроллер*;
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (*модели*), вообще не будут осведомлены о том, какое *представление* будет использоваться.

**8. Объясните принцип "Единственной ответственности" (Single Responsibility Principle, SRP), который используется бля проектирования и реализации независимого или слабосвязанного кода.**

Принцип обозначает, что каждый блок кода должен выполнять свои задачи, не перегружаясь.

**9. Что такое нисходящее и восходящее программирование?**

**Нисходящее программирование** - методика разработки программ, при которой разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой.

**Восходящее программирование** — методика разработки программ, при которой крупные блоки собираются из ранее созданных мелких блоков.

**10. В чём преимущество и для чего нужно использовать в C/C++ проектах файлы заголовки? Какие проблемы они решают? С помощью какой программной конструкции языка можно предотвратить бесконечное (рекурсивное) включение заголовочных файлов друг в друга?**

Есть ещё один тип файлов — **заголовочные файлы**(или ещё **«заголовки»**), которые имеют расширение**.h**. Целью заголовочных файлов является удобное хранение набора объявлений объектов для их последующего использования в других программах.

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 4

по дисциплине «***Операционные системы***»

тема: **«Автоматическая сборка многофайловых проектов»**

|  |  |  |
| --- | --- | --- |
| Исполнители: |  | студенты группы 10701118  Воробей И.А. |
| Преподаватель: |  | Станкевич С.Н. |

Минск 2020

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**Автоматическая сборка многофайловых проектов**

**Цель работы**

### Научиться эффективно использовать специальные средства для автоматизации процесса компиляции, сборки и запуска многофайловых проектов.

### Требования

1) Для автоматизации сборки проекта необходимо использовать стандартную оболочку сценариев Linux bash (/bin/sh), а в качестве утилиты автосборщика – GNU make.

2) Для автосборки проекта с использованием GNU make необходимо описать два типа make-файлов: простой и параметризированный.

3) При выполнении задания запрещается использовать IDE. Рекомендуется задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а для отладки – gdb-отладчик.

**Индивидуальное задание:**

Для программ, которые были разработаны в двух предыдущих лабораторных работах №2 и №3 необходимо добавить автоматизацию сборки многофайлового проекта с использованием сценарных оболочек и автосборщиков5) Среди столбцов заданной матрицы, содержащих только такие элементы, ко-торые по модулю не больше n, найти столбец с минимальным произведением элементов.

Ход выполнения лабораторной работы:

1. Установил соответствующий инструментарий для выполнения лабораторной работы (gcc, g++ и gdb).
2. Для удобства навигации по файлам установил дополнительно файловый менеджер GNOME Comander (gnome-commender).
3. Разработал алгоритм и блок-схему решения индивидуального задания.
4. В текстовом редакторе gedit набрал исходный код решения задания на языке программирования С++ и сохранил исходник с именем. Программа состоит из следующих файлов (см. рис. 1):
   * ***functions.h*** **–** файл заголовка, в котором описан прототип функции определения простого числа;
   * ***functions.cc*** **–** файл реализации функции определения простого числа;
   * ***test.cc*** **–** файл, в котором описана главная функция программы main**;**
5. Запустил терминал Linux и с помощью команды cd перешёл в папку местонахождения исходных кодов программы.
6. Для компиляции программы использовал компилятор g++ с вводом соответствующих параметров (см. рис. 2);
7. Для тестирования работоспособности запустил программу с соответствующими аргументами командной строки (см. рис. 2).
8. Для отладки программы использовал gdb (см. рис. 3).

Результаты выполнения лабораторной работы:

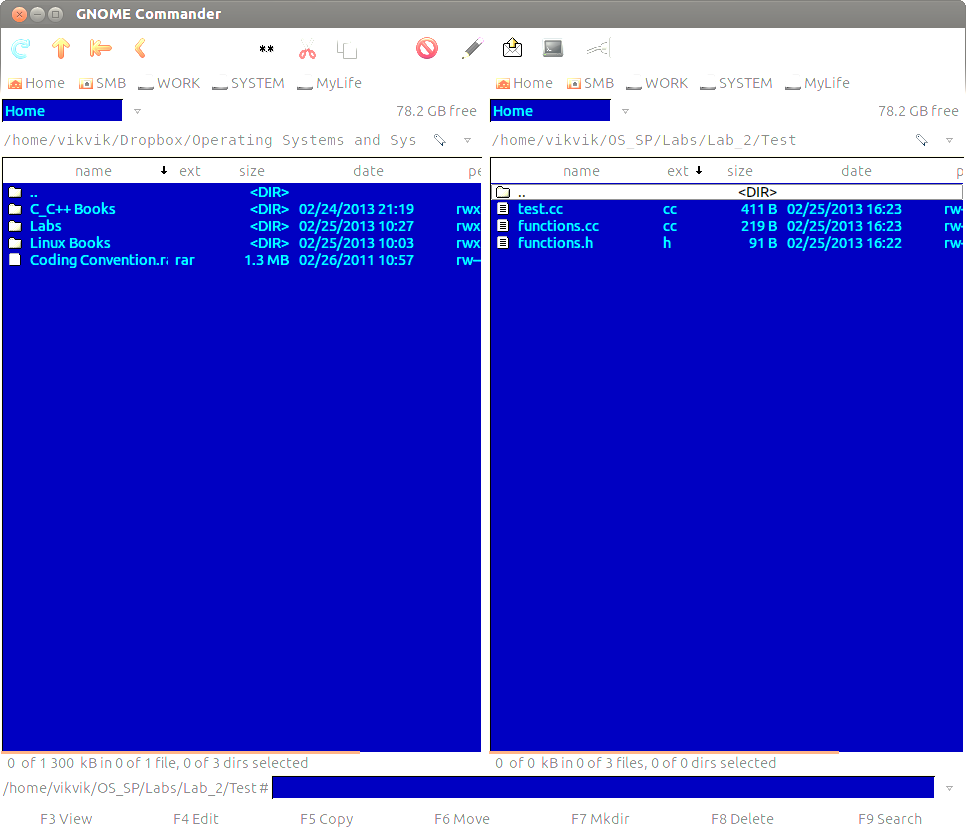


Рисунок 1 – Содержимое папки программы лабораторной работы № 2

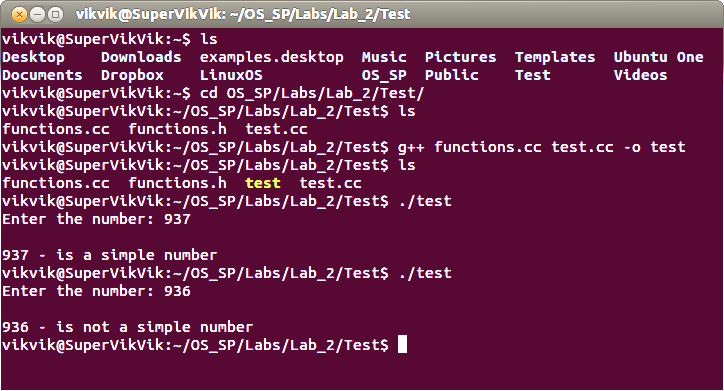


Рисунок 2 — Компиляция и выполнение программы test

Рисунок 3 — Запуск и отладка программы test с использованием gdb

**Контрольные вопросы:**

**1 Какие существуют способы и средства для автоматической сборки многофайловых проектов? Опишите категории утилит автосборок в Linux, а также их преимущества и недостатки?**

Утилиты make, make-file.

Утилита make автоматически определяет, какие части крупного проекта были изменены и должны быть перекомпилированы, а также выполняет действия, необходимые для этого.

По умолчанию имя файла сценария сборки - **Makefile**. Утилита make обеспечивает полную сборку указанной **цели,** присутствующей в сценарии.

**2. Как осуществить сборку и запуск проекта с использованием исполняемого файла стандартной скриптовой оболочки ОС Linux bash?**

#!/bin/bash  
#указываем где у нас хранится bash-интерпретатор  
parametr1=$1 #присваиваем переменной parametr1 значение первого параметра скрипта  
script\_name=$0 #присваиваем переменной script\_name значение имени скрипта  
echo "Вы запустили скрипт с именем $script\_name и параметром $parametr1" # команда echo выводит определенную строку, обращение к переменным осуществляется через $имя\_переменной.  
echo 'Вы запустили скрипт с именем $script\_name и параметром $parametr1' # здесь мы видим другие кавычки, разница в том, что в одинарных кавычках не происходит подстановки переменных.  
exit 0 #Выход с кодом 0 (удачное завершение работы скрипта)

**3. Зачем и где используется утилита GNU make (автосборщик)?**

Утилита make автоматически определяет какие части большой программы должны быть перекомпилированы, и выполняет необходимые для этого действия.

**4. Зачем нужен make-файл? Что он содержит?**

Для работы с утилитой make, вам понадобится так называемый **make-файл (makefile)**, который будет содержать описание требуемых действий. Как правило, make-файл описывает, каким образом нужно компилировать и компоновать программу.

**5. Опишите общий алгоритм процесса автоматической сборки программы.**

Включает такие действия, как:

* компиляция исходного кода в объектный модуль,
* сборка бинарного кода в исполняемый файл,
* выполнение тестов,
* развёртывание программы в целевой среде,
* написание сопроводительной документации или описание изменений новой версии.

**6. Опишите базовый синтаксис make-файла и общие правила определения целевых связок в make-файле: целей (targets), зависимостей (dependencies) и инструкций (instructions)? Какая цель будет обрабатываться самой первой при запуске утилиты make? Что такое «основная цель сборки» и что в неё входит?**

Базовый синтаксис:

*# Индентация осуществляется исключительно при помощи символов табуляции,*

*# каждой команде должен предшествовать отступ*

<цели>: <реквизиты>

<команда *#1>*

...

<команда *#n>*

Обычно, **цель (target)** представляет собой имя файла, который генерируется в процессе работы утилиты make. Примером могут служить объектные и исполняемый файлы собираемой программы. Цель также может быть именем некоторого действия, которое нужно выполнить.

**7. Какое необходимо дать стандартное имя make-файлу, чтобы его автоматически распознал автосборщик? А если в проекте используется нестандартное имя make-файла, как нужно вызывать утилиту make?**

По умолчанию, когда make ищет make-файл для обработки, она поочередно пробует найти файлы со следующими именами (в указанном порядке): `GNUmakefile', `makefile' и `Makefile'.

Если вы хотите использовать "нестандартное" имя для вашего make-файла, вы можете указать его в командной строке, используя опции `-f' или `--file'. Аргументы `-f имя\_файла' или `--file=имя\_файла', указывают программе make на необходимость использования файла с именем имя\_файла в качестве make-файла. Вы можете задать обработку сразу нескольких make-файлов, перечислив их в командной строке с помощь нескольких опций `-f' или `--file'. Все указанные таким образом make-файлы логически "объединяются" в том порядке, как они были заданы в командной строке. При наличии в командной строке опций `-f' или `--file', автоматического поиска make-файлов с именами `GNUmakefile', `makefile' и `Makefile', не производится.

**8. Параметризация процесса сборки и переменные make-файла?**

Если среди переменных среды (environment variables) имеется переменная с именем MAKEFILES, то ее содержимое интерпретируется как список имен (разделенных пробелами) дополнительных make-файлов, которые должны быть прочитаны перед тем, как начнут обрабатываться "основные" make-файлы. Этот механизм работает во многом аналогично директиве include. Аналогичным образом производится и поиск этих дополнительных make-файлов в разных каталогах (смотрите раздел [Подключение других make-файлов](http://linux.yaroslavl.ru/docs/prog/gnu_make_3-79_russian_manual.html#SEC15)). При этом, главная цель не может браться из этих файлов, а отсутствие какого-либо из них не рассматривается как ошибка.

**9. Зачем нужен рекурсивный вызов make-утилиты? Как его осуществить?**

При рекурсивном использовании, программа make сама выступает в качестве одной из команд make-файла. Подобная техника полезна, когда вы хотите иметь отдельные make-файлы для различных подсистем, составляющих большую систему. Предположим, у вас имеется подкаталог `subdir', содержащий свой собственный make-файл, и вы хотите, чтобы make-файл из "объемлющего" каталога запускал make в этом подкаталоге.

**10. Как избежать перекомпиляции проекта и при этом сделать все файлы исходного кода программы последними или «свежими» (up to date), т.е. не нуждающимися в перекомпиляции?**

Можно сделать параллельную или последовательную компиляцию, используя при этом правильную последовательность.

**11. Как генерировать случайные значения с использованием стандартных функций rand() и srand(…) из стандартной C-библиотеки stdlib? Приведите пример. Как искусственно с помощью данных функций задать диапазон генерирования случайных значений?**

uint32\_t bounded\_rand(rng\_t& rng, uint32\_t range)

{ std::uniform\_int\_distribution<uint32\_t> dist(0, range-1)

; return dist(rng); }

Пример генерации кода с использованием функций.

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №5

По дисциплине **«Операционные системы»**

Тема: **«Memory and Args»**

**Выполнил:**

Студент группы 10701118

Воробей

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Закрепить знания и практические навыки эффективного использования памяти при проектировании и реализации интерактивных приложений на языке программирования C/C++.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Входные данные для программы должны вводиться в виде набора строк! Если в задании сказано про «слова» – это значит, что вводится должна одна строка, которая состоит из несколько слов. Если сказано про «строки», то должен вводиться набор строк, каждая из которых состоит минимум из двух трёх слов.

3) Предусмотреть два способа инициализации данных: с помощью аргументов командной строки и пользовательского ввода (при запуске программа должна проверять наличие параметров и использовать их для выполнения своего алгоритма при их наличии, в противном случае, запрашивать соответствующие данные у пользователя).

4) Программа должна выводить на консоль исходные данные и конечный результат.

5) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

6) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

7) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам.

8) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

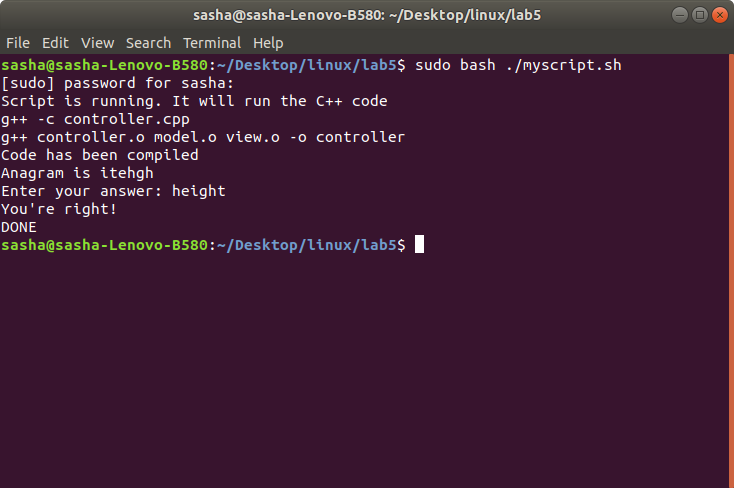
9) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++.

10) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make

**Основное задание:**

Написать программу «Анаграммы» (anagrams), которая реализует популярную словесную игру. Игра «Анаграммы» очень интересная и необычная игра в которой нет ни картинок ни слов. Суть игры (программы) заключается в следующем: формируется группа слов, компьютер случайным образом выбирает одно из слов и случайным образом переставляет в нём буквы, а затем представляет пользователю (игроку). Цель игрока – угадать выбранное компьютером слово.

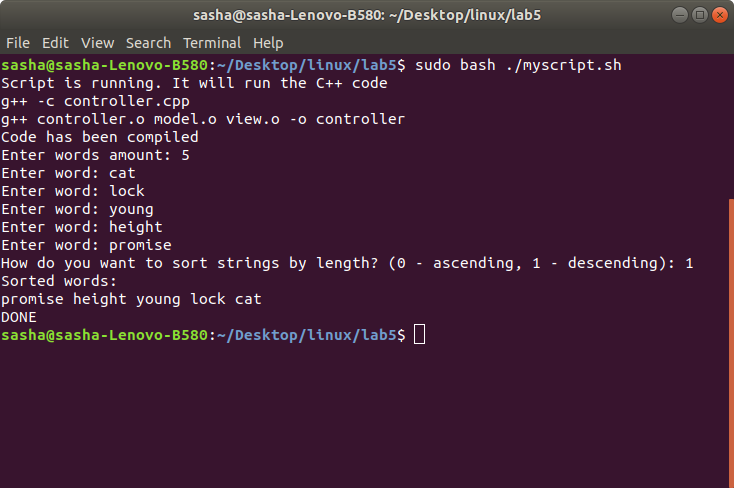
*Результаты выполнения*

**

**Индивидуальное задание:**

2) Упорядочить и вывести строки в порядке возрастания (убывания) значений их длины.

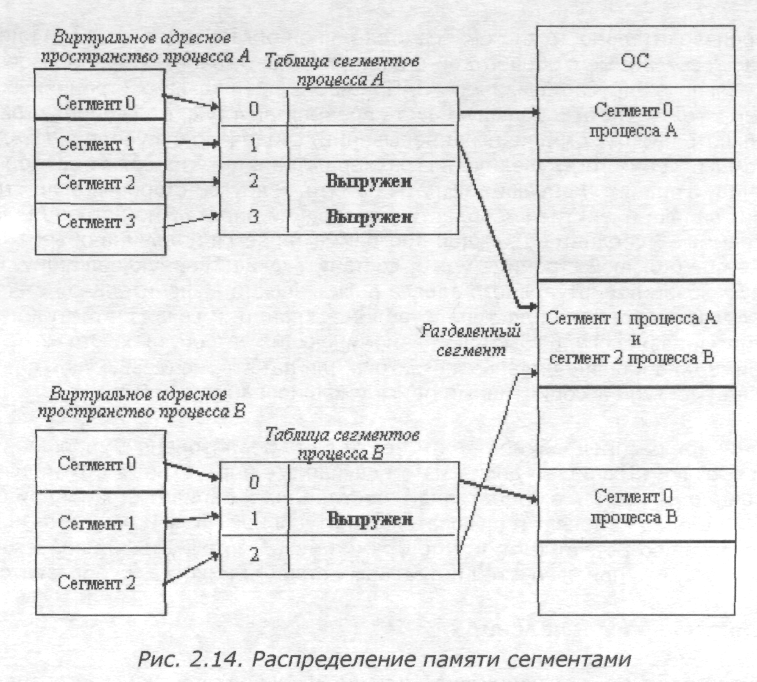
*Результаты выполнения*

**

**Контрольные вопросы:**

**1.** На какие укрупнённые блоки (области или сегменты) делиться виртуальное адресное пространство, которое выделяется операционной системой для каждой программы?

*Ответ:*



**2.** Для чего каждый из блоков нужен, что там хранится и в какой момент определяется его размер?

*Ответ:*

Размер пользовательского процесса ограничен объемом логического адресного пространства. Характерный размер логической памяти определяется разрядностью архитектуры и составляет для современных систем 232 (в недалеком будущем 264) *байт*. Эта величина обычно существенно превышает объем оперативной памяти, поэтому часть пользовательского процесса прозрачным образом может быть размещена во внешней памяти. Поэтому у пользователя создается иллюзия того, что он имеет дело с виртуальной памятью, отличной от реальной, размер которой потенциально больше, чем размер оперативной памяти. В дальнейшем наряду с термином "*логическая память*" будет употребляться термин "*виртуальная память*".

Для определения схемы виртуальной памяти, реализованной в ОС *Windows*, лучше всего подходит термин "сегментно-*страничная виртуальная память*"

**3.** Что такое класс памяти (storage class) переменной и что он определяет?

*Ответ:*

Класс памяти переменной (англ. Storage class) — понятие в некоторых языках программирования. Он определяет область видимости переменной, а также как долго переменная находится в памяти.

**4**. Какие существуют разновидности классов памяти в языках программирования С/С++ и их основные отличия?

*Ответ:*

* Автоматический (automatic);
* Регистровый (register);
* Статический (static);
* Внешний (external).

АВТОМАТИЧЕСКИЕ

По умолчанию, локальные переменные имеют класс auto. Такие переменные располагаются на стеке, а их область видимости ограничена своим блоком. Спецификатор auto уведомляет компилятор о том, что локальная переменная, перед именем которой он стоит, создаётся при входе в блок и разрушается при выходе из блока. Все переменные, определённые внутри функции, являются автоматическими по умолчанию, и поэтому ключевое слово auto используется крайне редко.

РЕГИСТРОВЫЙ

Когда мы определяем регистровую переменную, то мы просим компилятор, чтобы переменная располагалась в регистре, а не в оперативной памяти. Компилятор может сделать переменную регистровой, если позволяют условия (регистры не заняты, и по мнению компилятора это не приведёт к увеличению издержек). Регистровые переменные определяются с помощью служебного слово register перед типом.

СТАТИЧЕСКИЙ

Спецификатор static указывает компилятору на хранение локальной переменной во время всего жизненного цикла программы вместо ее создания и разрушения при каждом входе в область действия и выходе из неё. Следовательно, возведение локальных переменных в ранг статистических позволяет поддерживать их значения между вызовами функций.

Модификатор static можно также применять к глобальным переменным. В этом случае область видимости такой переменной ограничивается файлом, в котором она объявлены, это означает, что переменная будет иметь внутреннюю привязку. Внутренняя привязка говорит о том, что индикатор известен только внутри своего файла. Если функция объявлена как static, то она видна только в своём файле. Из другого файла к static функции обратиться нельзя.

ВНЕШНИЙ

Внешние переменные вводятся как противоположные автоматическим. Это глобальные переменные и к ним можно обращаться именами из любой функции. Поскольку внешние переменные доступны везде, их можно использовать для связи между функциями, не пренебрегая механизму формальных параметров.

Класс памяти extern в Си используем в двух случаях:

* если переменная объявляется в программе ниже, чем ссылка на неё;
* если переменная объявлена в другом модуле.

**5**. Какие есть классы памяти у переменных? Какой из классов памяти и при каких условия используется по умолчанию (неявно) при объявлении переменной?

*Ответ:*

Классы памяти переменных. По умолчанию,локальные переменные имеют класс auto. Такие переменные располагаются на стеке а их область видимости ограничена своим блоком

* Автоматический (automatic);
* Регистровый (register);
* Статический (static);
* Внешний (external).

**6**. Какие есть классы памяти у функций? Какой из классов памяти и при каких условия используется по умолчанию (неявно) при описании функции?

*Ответ:*

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных  из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию.

В основном нужны для работы с динамическими массивами и для передачи данных в функции.

**7**. Что такое указатели и зачем они нужны при программировании?

*Ответ:*

**8**. Опишите основные операторы (операции), которые применяются для работы с указателями.

*Ответ:*

Ссылки – особый тип данных, являющийся скрытой формой указателя, который при использовании автоматически разименовывается.

1. Нельзя объявить массив ссылок.
2. У ссылки нет адреса.
3. Существует арифметика указателей, но нет арифметики ссылок.
4. Указатель может иметь «невалидное» значение с которым его можно сравнить перед использованием.

**9.** Что такое ссылка и чем указатели отличаются от ссылок в языке программирования С++?

*Ответ:*

**10.** Для чего необходимо использовать динамическое распределение памяти?

*Ответ:*

Динамическое распределение памяти — способ выделения оперативной памятикомпьютера для объектов в программе, при котором выделение памяти под объект осуществляется во время выполнения программы.

**11.** Опишите основные функции стандартной библиотеки языка программирования С и особенность их использования для работы с динамической памятью: malloc, calloc, realloc, free.

*Ответ:*

|  | |
| --- | --- |
| |  |  | | --- | --- | | 1 | void\* malloc(size\_t size); | |  |

Выделяет блок памяти размером size.

|  | |
| --- | --- |
| |  |  | | --- | --- | | 1 | void\* calloc(size\_t num, size\_t size); | |  |

Выделяет блок памяти для массива из num эелементов, где каждый элемент имеет размер равный size и выставляет все биты блока памяти в 0. То есть по сути выделяет проинициализированный нулем блок размера num \* size.

|  | |
| --- | --- |
| |  |  | | --- | --- | | 1 | void\* realloc(void\* ptr, size\_t size); | |  |

**12.** Опишите основные операторы языка программирования С++ для работы с динамической памятью: new и delete.

*Ответ:*

New – оператор выделяющий область в хип-памяти под объект, а также запускает конструктор объекта.

Delete – удаление объектов и чистка памяти.

**13.** Какие бывают ошибки при работе с динамической памятью?

*Ответ:*

1. ошибка связанна с возможным переполнением динамической области памяти
2. утечка памяти
3. повторный возврат памяти
4. нулевой указатель

**14**. Какие существуют способы передачи параметров в функцию в языках программирования С/C++?

*Ответ:*

- передача по значению

- передача по адресу(указатель)

- передача по ссылке

**15**. Что происходит с оригинальной переменной, передаваемой в качестве параметра в функцию, и памятью, выделенной для программы, в момент вызова соответствующей функции?

*Ответ:*

Создается копия в новом фрагменте памяти.

**16**. Что такое «утечка памяти» (memory leak)? Когда, где и по чьей вине это может возникнуть?

*Ответ:*

Утечка памяти (memory leak) - это процесс неконтролируемого уменьшения объёма свободной оперативной или виртуальной памяти компьютера, связанный с ошибками в работающих программах, вовремя не освобождающих ненужные уже участки памяти.

Происходит по вине программиста. Однако в более молодых языках присутствует автосборщик мусора, и утечка памяти практически невозможна.

**17**. Как создать одномерный массив конкретного типа данных? Как объявить массив, который может хранить элементы различных типов данных?

*Ответ:*

Type name[amount];

Хранение разных типов данных в массивах доступно не во всех языках, для этого лучше воспользоваться коллекциями. Например vector в с++.

**18**. Какие существуют способы для работать с одномерными динамическими массивами?

*Ответ:*

Для работы динамических массивов нам понадобится при инициализации [указатель](https://codelessons.ru/cplusplus/ukazateli-v-c-podrobnoe-rukovodstvo.html)(всего лишь при инициализации!) и уже знакомый конструктор new.

**19**. Опишите способы динамического выделения и освобождения памяти для многомерного массива (минимум три)? Какой из способов эффективней и почему?

*Ответ:*

int \*p;

int sizeof(тип);

// Выделение памяти под указатели на строки  
  a = (int\*\*)malloc(n \* sizeof(int\*));

// Выделение памяти  
  a = (int\*)malloc(n\*m \* sizeof(int));

a = (int\*\*)malloc(n \* sizeof(int\*));  
  m = (int\*)malloc(n \* sizeof(int)); // массив кол-ва элеменов в строках массива

**20**. Какие существуют способы для работать с многомерными динамическими массивами

*Ответ:*

Работа с динамическим многомерным массивом синтаксически полностью совпадает с работой с многомерным массивом.

**Листинг:**

1. **controler.cpp**

#include <iostream>

#include <string>

#include "model.h"

#include "view.h"

using namespace std;

void executeMainTask();

void executeIndividualTask();

int main()

{

//executeMainTask();

executeIndividualTask();

return 0;

}

void executeMainTask()

{

string anagram = generateAndShuffleRandomWord();

cout << "Anagram is " << anagram << endl;

string word;

cout << "Enter your answer: ";

cin >> word;

bool result = guessAnagram(word, anagram);

printAnagramResult(result);

}

void executeIndividualTask()

{

int n;

cout << "Enter words amount: ";

cin >> n;

string\* words = new string[n];

for(int i = 0; i < n; i++)

{

cout << "Enter word: ";

cin >> words[i];

}

cout << "How do you want to sort strings by length? (0 - ascending, 1 - descending): ";

int choice;

cin >> choice;

if(choice == 0)

{

sortArray(words, n, true);

} else

{

sortArray(words, n, false);

}

cout << "Sorted words:" << endl;

printStrings(words, n);

delete[] words;

}

1. **model.cpp**

#include <cstdlib>

#include <string>

#include <algorithm>

#include "model.h"

using namespace std;

string\* getWords()

{

return new string[5] {"cat", "dog", "computer", "height", "weight"};

}

string generateAndShuffleRandomWord()

{

int randomIndex = rand() % 4;

string\* words = getWords();

string word = words[randomIndex];

random\_shuffle(word.begin(), word.end());

delete[] words;

return word;

}

bool areAnagrams(string word, string anagram)

{

int n1 = word.length();

int n2 = anagram.length();

if (n1 != n2)

return false;

sort(word.begin(), word.end());

sort(anagram.begin(), anagram.end());

for (int i = 0; i < n1; i++)

if (word[i] != anagram[i])

return false;

return true;

}

bool guessAnagram(string word, string anagram)

{

string\* words = new string[5] {"cat", "dog", "computer", "height", "weight"};

string guessedWord;

for(int i = 0; i < 5; i++)

{

if(areAnagrams(words[i], anagram))

{

guessedWord = words[i];

}

}

delete[] words;

return guessedWord == word;

}

bool matches(string first, string second, bool ascending)

{

if(ascending)

{

return first.length() > second.length();

} else

{

return first.length() < second.length();

}

}

void sortArray(string\* arr, int size, bool ascending)

{

for(int i = 0; i < size - 1; i++)

{

for(int j = 0; j < size - i - 1; j++)

{

if(matches(arr[j], arr[j + 1], ascending))

{

string temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

1. **model.h**

#include <string>

using namespace std;

#pragma once

string generateAndShuffleRandomWord();

bool guessAnagram(string word, string anagram);

void sortArray(string\* arr, int size, bool ascending);

1. **view.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printAnagramResult(bool result)

{

if(result)

{

cout << "You're right!" << endl;

} else

{

cout << "You're wrong!" << endl;

}

}

void printStrings(string\* arr, int size)

{

for(int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

1. **view.h**

#include <string>

#pragma once

void printAnagramResult(bool);

void printStrings(std::string\* arr, int size);

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №6

По дисциплине **«Операционные системы»**

Тема: **«Libraries»**

**Выполнил:**

Студент группы 10701118

Воробей

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Изучить и закрепить на практике создание и использование статически и динамически подключаемых библиотек в операционных системах семейства Linux.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Входные данные для программы должны вводиться в виде набора строк! Если в задании сказано про «слова» – это значит, что вводится должна одна строка, которая состоит из несколько слов. Если сказано про «строки», то должен вводиться набор строк, каждая из которых состоит минимум из двух трёх слов.

3) Предусмотреть два способа инициализации данных: с помощью аргументов командной строки и пользовательского ввода (при запуске программа должна проверять наличие параметров и использовать их для выполнения своего алгоритма при их наличии, в противном случае, запрашивать соответствующие данные у пользователя).

4) Программа должна выводить на консоль исходные данные и конечный результат.

5) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

6) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

7) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

8) Предусмотреть создание двух типов библиотек: статической и динамической. 9) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

10) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

11) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

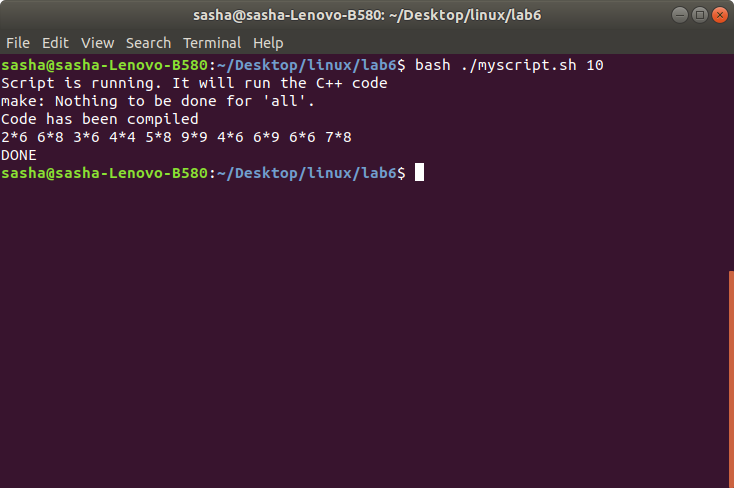
12) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

13) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Для проверки остаточных знаний учеников после рождественских каникул, учитель младших классов решил начинать каждый урок с того, чтобы задавать каждому ученику пример из таблицы умножения, но в классе, к примеру, 15 человек, а примеры среди них не должны повторяться. В помощь учителю разработайте универсальную библиотечную функцию, которая будет выводить на экран 15 (или N) случайных примеров из таблицы умножения (к примеру, от 2\*2 до 9\*9, потому что задания по умножению на 1 и на 10 – слишком просты). При этом среди 15 примеров не должно быть повторяющихся (примеры 2\*3 и 3\*2 и им подобные пары считать повторяющимися). Алгоритм поиска данных комбинаций должен быть эффективным!!! Напишите тестовую программу, которая бы демонстрировала работоспособность данной библиотечной функции.

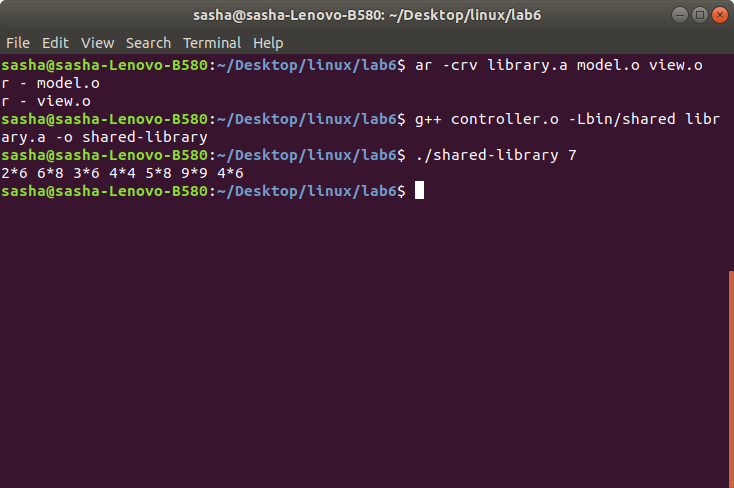
*Результаты выполнения*



**Индивидуальное задание:**

Произвести рефакторинг предыдущей лабораторной работы: вынесите код функций бизнес логики в отдельную библиотеку.

*Результаты выполнения*

****

**Контрольные вопросы:**

**1.** Что такое библиотеки и для чего они используются?

*Ответ:*

Библиотеки - это часть готового функционала, который используется большим количеством программистов.  
Чтобы не писать тот функционал, который всем нужен и который уже реализован и протестирован

**2.** Опишите существующие типы библиотек и принципы их использования.

*Ответ:*

Библиотеки делятся на статические и динамические.

|  |
| --- |
| Ввод-вывод:   * [fstream](https://ru.wikipedia.org/wiki/Fstream) * iomanip * [ios](https://ru.wikipedia.org/w/index.php?title=Ios_(C%2B%2B)&action=edit&redlink=1) * iosfwd * [iostream](https://ru.wikipedia.org/wiki/Iostream) (ostream, istream) * [sstream](https://ru.wikipedia.org/wiki/Sstream) * streambuf   [Многопоточность](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C):   * atomic (C++11) * condition\_variable (C++11) * future (C++11) * mutex (C++11) * shared\_mutex (C++14) * thread (C++11)   Прочие:   * [algorithm](https://ru.wikipedia.org/wiki/Algorithm_(C%2B%2B)) * [bitset](https://ru.wikipedia.org/w/index.php?title=Bitset_(C%2B%2B)&action=edit&redlink=1) * chrono (C++11) * codecvt (C++11) * complex * exception * [functional](https://ru.wikipedia.org/wiki/Functional_(C%2B%2B)) * initializer\_list (C++11) * [iterator](https://ru.wikipedia.org/w/index.php?title=%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B_%D0%B2_C%2B%2B&action=edit&redlink=1) * limits * locale * memory * [new](https://ru.wikipedia.org/wiki/New_(C%2B%2B)) * numeric * random (C++11) * ratio (C++11) * regex (C++11) * stdexcept * [string](https://ru.wikipedia.org/wiki/String_(C%2B%2B)) * system\_error (C++11) * tuple (C++11) * typeindex (C++11) * typeinfo * type\_traits (C++11) * utility * valarray |
| [**Стандартная библиотека шаблонов**](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%BE%D0%B2) |
| [Последовательные контейнеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80):   * [array](https://ru.wikipedia.org/w/index.php?title=Array_(C%2B%2B)&action=edit&redlink=1) (C++11) * [deque](https://ru.wikipedia.org/w/index.php?title=Deque_(C%2B%2B)&action=edit&redlink=1) * [forward\_list](https://ru.wikipedia.org/w/index.php?title=Forward_list&action=edit&redlink=1) (C++11) * [list](https://ru.wikipedia.org/w/index.php?title=List_(C%2B%2B)&action=edit&redlink=1) * [vector](https://ru.wikipedia.org/wiki/Vector_(C%2B%2B)), vector<bool>   Ассоциативные контейнеры:   * [map](https://ru.wikipedia.org/wiki/Map_(C%2B%2B)) * [set](https://ru.wikipedia.org/w/index.php?title=Set_(C%2B%2B)&action=edit&redlink=1) * [multimap](https://ru.wikipedia.org/w/index.php?title=Multimap_(C%2B%2B)&action=edit&redlink=1) * [multiset](https://ru.wikipedia.org/w/index.php?title=Set_(C%2B%2B)&action=edit&redlink=1) * [unordered\_map](https://ru.wikipedia.org/w/index.php?title=Unordered_map_(C%2B%2B)&action=edit&redlink=1) (C++11) * [unordered\_set](https://ru.wikipedia.org/w/index.php?title=Unordered_set&action=edit&redlink=1) (C++11) * [unordered\_multimap](https://ru.wikipedia.org/w/index.php?title=Unordered_map_(C%2B%2B)&action=edit&redlink=1) (C++11) * [unordered\_multiset](https://ru.wikipedia.org/w/index.php?title=Unordered_set&action=edit&redlink=1) (C++11)   Контейнеры-адаптеры:   * [stack](https://ru.wikipedia.org/w/index.php?title=Stack_(C%2B%2B)&action=edit&redlink=1) * priority\_queue * [queue](https://ru.wikipedia.org/w/index.php?title=Queue_(C%2B%2B)&action=edit&redlink=1) |
| [**Стандартная библиотека языка программирования Си**](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0_%D0%A1%D0%B8) |
| * [cassert](https://ru.wikipedia.org/wiki/Assert.h) * [cctype](https://ru.wikipedia.org/wiki/Ctype.h) * [cerrno](https://ru.wikipedia.org/wiki/Errno.h) * [cfenv](https://ru.wikipedia.org/wiki/Fenv.h) (C++11) * [cfloat](https://ru.wikipedia.org/wiki/Float.h) * [cinttypes](https://ru.wikipedia.org/wiki/Inttypes.h) (C++11) * [ciso646](https://ru.wikipedia.org/wiki/Iso646.h) * [climits](https://ru.wikipedia.org/wiki/Limits.h) * [clocale](https://ru.wikipedia.org/wiki/Locale.h) * [cmath](https://ru.wikipedia.org/wiki/Math.h) * [csetjmp](https://ru.wikipedia.org/wiki/Setjmp.h) * [csignal](https://ru.wikipedia.org/wiki/Signal.h) * [cstdarg](https://ru.wikipedia.org/wiki/Stdarg.h) * [cstdbool](https://ru.wikipedia.org/wiki/Stdbool.h) (C++11) * [cstddef](https://ru.wikipedia.org/wiki/Stddef.h) * [cstdint](https://ru.wikipedia.org/wiki/Stdint.h) (C++11) * [cstdio](https://ru.wikipedia.org/wiki/Stdio.h) * [cstdlib](https://ru.wikipedia.org/wiki/Stdlib.h) * [cstring](https://ru.wikipedia.org/wiki/String.h) * [ctgmath](https://ru.wikipedia.org/w/index.php?title=Tgmath.h&action=edit&redlink=1) (C++11) * [ctime](https://ru.wikipedia.org/wiki/Time.h) * [cuchar](https://ru.wikipedia.org/w/index.php?title=Uchar.h&action=edit&redlink=1) (C++11) * [cwchar](https://ru.wikipedia.org/wiki/Wchar.h) * [cwctype](https://ru.wikipedia.org/wiki/Wctype.h) |

**3.** Опишите преимущества и недостатки каждого из типа библиотек.

*Ответ:*

Статическая библиотека это фактически архив объектных файлов, который используется в процессе статической линковки. В результате статической линковки из многих объектных файлов получается один исполняемый, запускается статическая линковка в момент создания исполняемого файла.  
  
Динамическая библиотека это фактически исполняемый файл (т.е. DLL и EXE в Windows имеют одинаковый формат). Динамическая линковка запускается в момент создания процесса (когда вы запускаете исполняемый файл на выполнение), линкуются между собой несколько исполняемых файлов каждый раз, когда создается новый процесс. Так же возможна динамическая линковка уже после запуска, т.е. новая библиотека может быть подгружена в адресное пространство ужа работающего процесса.

**4.** Как подключить библиотеку к программе? Опишите специфику подключения.

*Ответ:*

Так

#include "stdlib.h"

Или так

#include <iostream>

**5.** На какой стадии происходит подключение библиотек (объектных файлов) к основному модулю программы?

*Ответ:*

Подключение библиотекипроисходит на этапе компоновки исполняемого файла.

**6.** Чем отличается заголовочные файлы (файлы с расширением \*.h) от файлов библиотек?

*Ответ:*

Конкретно, .lib (.a), .dll - библиотеки: упакованные определённым образом реализации (функций, методов и пр.). Чтобы их использовать нужны заголовочные файлы, в которых, обычно, содержатся прототипы (интерфейсы) функций, методов и пр.

**7.** Как создать статическую (динамическую) библиотеку и подключить её к программе?

*Ответ:*

-ar –crv lib\_mae.a FILES.o

Link with main

Создание проекта **статической** библиотеки в Visual Studio 2019

1. В строке меню последовательно выберите файл > создать > проект , чтобы открыть диалоговое окно Создание нового проекта .
2. В верхней части диалогового окна для параметра Язык установите значение C++ , для параметра Платформа — значение Windows, а для параметра Тип проекта — значение Библиотека.
3. В отфильтрованном списке типов проектов выберите статическая библиотека , а затем нажмите кнопку Далее. На следующей странице введите MathFuncsLib в поле имя , чтобы указать имя проекта, и при необходимости укажите расположение проекта.
4. Нажмите кнопку Создать, чтобы создать клиентский проект.

### Создание проекта библиотеки **DLL** в Visual Studio 2019

1. В строке меню последовательно выберите **файл > создать > проект** , чтобы открыть диалоговое окно **Создание нового проекта** .
2. В верхней части диалогового окна для параметра **Язык** установите значение **C++** , для параметра **Платформа**— значение **Windows,** а для параметра **Тип проекта** — значение **Библиотека.**
3. В отфильтрованном списке типов проектов выберите **Библиотека динамической компоновки (DLL)**, а затем нажмите кнопку **Далее.**
4. На странице **Настройка нового проекта** введите маслибрари в поле **имя проекта** , чтобы указать имя проекта. Оставьте значения **расположения** и **имени решения** по умолчанию. Задайте **решение**для**создания нового решения**. Снимите флажок **размещать решение и проект в том же каталоге,** если он установлен.
5. Нажмите кнопку **Создать,** чтобы создать проект.

**8.** Что такое «позиционно-независимый код» (position independent code, PIC)?

*Ответ:*

**Позиционно-независимый код** (англ. position-independent code) — программа, которая может быть размещена в любой области памяти, так как все ссылки на ячейки памяти в ней относительные (например, относительно счётчика команд). Такую программу можно переместить в другую область памяти в любой момент, в отличие от перемещаемой программы, которая хотя и может быть загружена в любую область памяти, но после загрузки должна оставаться на том же месте.

**9**. Какие способы можно использовать для того, чтобы указать местоположение подключаемых динамических библиотек?

*Ответ:*

По имени библиотеке

По полному пути к библиотеке

**10**. Если в целевом каталоге присутствует два типа одной и той же библиотеки, какая из них всегда будет подключаться по умолчанию?

*Ответ:*

По умолчанию будет подключаться первая(новая) библиотека.

**Листинг:**

* 1. **Controller.cpp**

#include <iostream>

#include <string>

#include "model.h"

#include "view.h"

using namespace std;

int main(int argc, char \*argv[])

{

if(argc < 2)

cout << "Invalid args" << endl;

int amount = stoi(argv[1]);

string\* pairs = generatePairs(amount);

printStrings(pairs, amount);

delete[] pairs;

return 0;

}

* 1. **Model.cpp**

#include <string>

#include <algorithm>

#include <bits/stdc++.h>

#include <iostream>

#include "model.h"

using namespace std;

string\* generatePairs(int amount)

{

const int size = 36;

string\* allPairs = new string[size];

int counter = 0;

for(int first = 2; first < 10; first++)

{

for(int second = first; second < 10; second++)

{

allPairs[counter] = to\_string(first) + "\*" + to\_string(second);

counter++;

}

}

random\_shuffle(allPairs, allPairs + size);

string\* pairs = new string[amount];

for(int i = 0; i < amount; i++)

{

pairs[i] = allPairs[i];

}

delete[] allPairs;

return pairs;

}

* 1. **Model.h**

#include <string>

using namespace std;

#pragma once

string\* generatePairs(int amount);

* 1. **View.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printStrings(string\* arr, int size)

{

for(int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

* 1. **View.h**

#include <string>

#pragma once

void printStrings(std::string\* arr, int size);

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №7

По дисциплине **«Операционные системы»**

Тема: **«IO with C»**

**Выполнили:**

Студенты группы 10701118

Воробей

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Изучить фундаментальные основы и концепции файлового ввода-вывода в Linux, основные функции ввода-вывода стандартной библиотеки языка программирования С, а также закрепить навыки применения данной библиотеки при написании системных программ.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Предусмотреть ещё один способ инициализации данных и вывода результата работы программы – с помощью файлового ввода-вывода. Программа должна дополнительно выводить на консоль исходные данные и конечный результат.

3) Функции бизнес логики и ввода-вывода должны быть сгруппированы и вынесены в соответствующие библиотеки. Рекомендуется использовать динамические библиотеки.

4) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

5) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

6) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

7) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

8) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

9) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

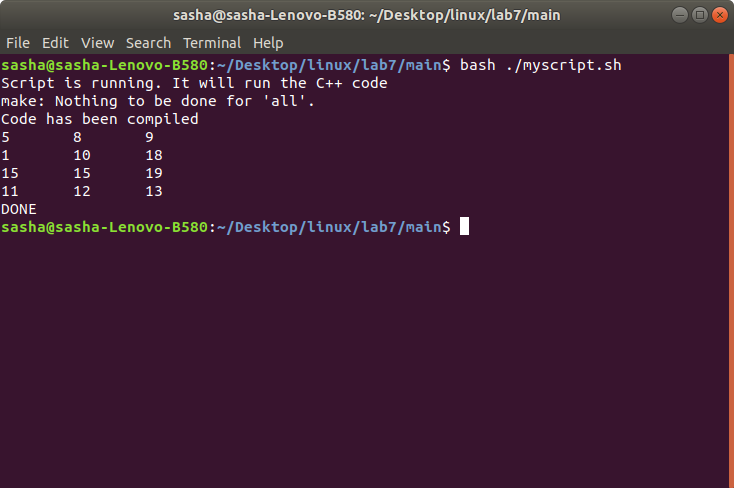
10) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

11) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Необходимо расширить функционал любой из программ, созданных в предыдущих лабораторных работах: возможность задавать размеры матрицы и её элементы с помощью указанного файла; возможность выводить результат в файл. Входной файл может содержать совокупность строк. Каждая строка файла содержит элементы матрицы. Необходимо предусмотреть как ручное создание файла и его заполнение, так и генерацию файла в автоматическом режиме.

*Результаты выполнения*

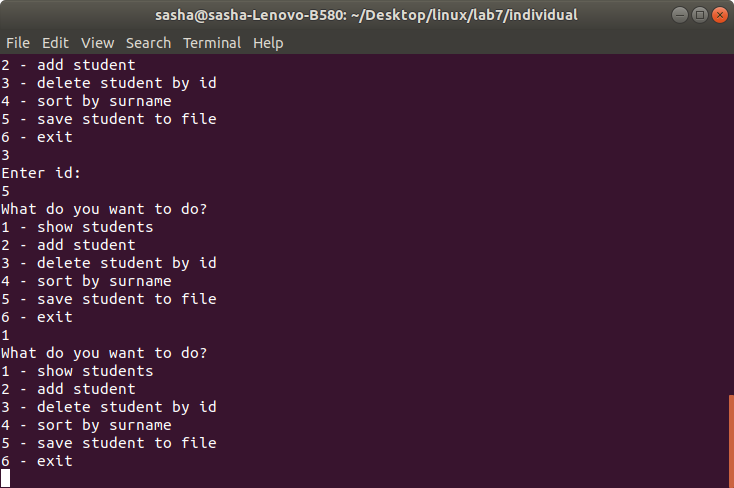


**Индивидуальное задание:**

В программе необходимо определить структуру сущностей согласно варианту задания (разрешается придумать свою структуру-сущность, предварительно согласовав её с преподавателем). Затем необходимо объявить динамическую структуру данных для хранения переменных данного структурного типа. Далее программа должна предоставлять возможность манипулировать данными элементами (к примеру, осуществить инициализацию контейнера, добавление или удаление элементов, отсортировать контейнер элементов по заданному критерию, осуществить необходимы поиск, подсчёт чего-то, сохранить данные контейнера и т.д.). Для сохранения состояния контейнера и его восстановления использовать файловый ввод-вывод.

1. Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс, Группа.

*Результаты выполнения*



**Контрольные вопросы:**

**1**. Устройство и структура жёсткого диска? Что такое главная загрузочная запись (Master Boot Record, MBR)? Что такое загрузочный сектор (Boot Record, BR)?

*Ответ:*

Любой жёсткий диск можно представить как огромный «чистый лист», на который можно записывать данные и откуда потом их можно считать. Чтобы ориентироваться на диске, всё его пространство разбивают на небольшие «клеточки» — **сектора.** Сектор — это минимальная единица хранения данных на диске, обычно его размер составляет 512 байт. Все сектора на диске нумеруются: каждый из n секторов получает номер от 0 до n–1. Благодаря этому любая информация, записанная на диск, получает точный адрес — номера соответствующих секторов. Так что диск ещё можно представить как очень длинную строчку (ленточку) из секторов. Можете посчитать, сколько секторов на вашем диске размером в N гигабайт.

Представлять жёсткий диск как единый «лист» не всегда бывает удобно: иногда полезно «разрезать» его на несколько независимых листов, на каждом из которых можно писать и стирать что угодно, не опасаясь повредить написанное на других листах. Логичнее всего записывать раздельно данные большей и меньшей важности или просто относящиеся к разным вещам.

Конечно, над жёстким диском следует производить не физическое, а логическое разрезание, для этого вводится понятие раздел (partition). Вся последовательность (очень длинная ленточка) секторов разрезается на несколько частей, каждая часть становится отдельным разделом. Фактически, нам не придётся ничего разрезать (да и вряд ли бы это удалось), достаточно объявить, после каких секторов на диске находятся границы разделов.

Главная загрузочная запись (англ. master boot record, MBR) — код и данные, необходимые для последующей загрузки операционной системы и расположенные в первых физических секторах (чаще всего в самом первом) на жёстком диске или другом устройстве хранения информации.

Загрузочный сектор — это особый сектор на жёстком диске, дискете или другом дисковом устройстве хранения информации. (Для дискеты это первый физический сектор, для жесткого диска — первый физический сектор для каждого раздела) В процессе загрузки компьютера с дискеты он загружается в память программой POST (в компьютерах архитектуры IBM PC обычно с адреса 0000:7c00), ему передается управление командой long jump.

**2.** Последовательность действий от включения компьютера до старта ОС?

*Ответ:*

После включения [персонального компьютера (ПК)](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) на базе [процессора](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80)семейства [Intel 80x86](https://ru.wikipedia.org/wiki/X86), его процессор начинает работу в [реальном режиме](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%B5%D0%B6%D0%B8%D0%BC)адресации с сегментной организацией и выполнение инструкций процессора с адреса FFFF:0, инициализированного в паре регистров CS:IP (Code Segment : Instruction Pointer) после снятия сигнала RESET. В конце доступного процессору адресного пространства [оперативной памяти](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) из [ПЗУ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B7%D0%B0%D0%BF%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) [материнской платы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%BD%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0) загружается загрузчик базовой системы ввода-вывода ([BIOS](https://ru.wikipedia.org/wiki/BIOS)), на который передается выполнение по инструкции безусловного перехода, отображенной по адресу FFFF:0.

Программа инициализации [BIOS](https://ru.wikipedia.org/wiki/BIOS) с помощью программы [POST](https://ru.wikipedia.org/wiki/POST_(%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) проверяет, что устройства компьютера работают корректно и инициализирует их.

Затем BIOS опрашивает по порядку устройства из заранее сформированного (в консольной утилите CMOS SETUP) списка загрузочных устройств, пока не найдет загрузочное устройство. Если такое устройство не будет найдено, BIOS выведет на [консоль](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C) ПК сообщение об ошибке и процесс загрузки будет остановлен. При обнаружении BIOS’ом загрузочного устройства, с последнего будет произведена попытка чтения данных его нулевого сектора (512 байт, в которых должна находиться [загрузочная запись](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C)). После успешного завершения чтения данных нулевого сектора загрузочного устройства в оперативную память по начальному адресу 0000:7С00, BIOS передает на этот адрес дальнейшее исполнение инструкций процессора.

Нулевой сектор загрузочного [жесткого диска](https://ru.wikipedia.org/wiki/%D0%96%D0%B5%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) содержит так называемую [«Главную загрузочную запись» (MBR — Master Boot Record)](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C), содержащую как данные первых четырёх записей [таблицы разделов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%BE%D0%B2), так и инструкции процессора Intel 80x86, обеспечивающие поиск по этим записям активного раздела, с которого, посредством одной из функций базовой системы ввода-вывода — [INT 13h](https://ru.wikipedia.org/w/index.php?title=INT_13h&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [Int 13h](https://en.wikipedia.org/wiki/Int_13h))[[1]](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0#cite_note-1), реализованной в виде обработчика [прерывания](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) процессора, будет считана [загрузочная запись операционной системы](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D1%87%D0%B8%D0%BA_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B). Этот загрузочный сектор, как правило, зависит от операционной системы и должен произвести загрузку в оперативную память [ядра операционной системы](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) с последующей передачей на него исполнения инструкций процессора. Если активного раздела не существует, или [загрузочный сектор](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%81%D0%B5%D0%BA%D1%82%D0%BE%D1%80) активного раздела некорректен, MBR может загрузить резервный начальный загрузчик и передать управление ему. Резервный начальный загрузчик должен выбрать раздел (зачастую с помощью пользователя), загрузить его загрузочный сектор и передать ему управление, который, в свою очередь, передаёт управление непосредственно [загрузчику ОС](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D1%87%D0%B8%D0%BA_%D0%9E%D0%A1) (дальнейшие этапы загрузки зависят от типа/версии установленной [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%A1)).

**3.** Что такое файл (File)?

*Ответ:*

Файл (англ. file) — именованная область данных на носителе информации.

**4.** Что такое файловая система (File System) и зачем она нужна?

*Ответ:*

Файловая система - это часть ОС, которая отвечает за размещение, удаление, хранение информации на носителях (жестких дисках, дискетах, оптических носителях, flash-картах и т.д.), предоставление пользователям и приложениям доступа к этой информации, обеспечение ее безопасного совместного использования, а также защиту и восстановление при программных и аппаратных сбоях.

**5.** Основные цели файловой системы?

*Ответ:*

Целииспользования файла :

- *Долговременное и надежное хранение информации*. Долговременность достигается за счет использования запоминающих устройств, не зависящих от питания, а высокая надежность определяется средствами защиты доступа к файлам и общей организацией программного кода ОС, при которой сбои аппаратуры чаще всего не разрушают информацию, хранящуюся в файлах.

- *Совместное использование информации*. Файлы обеспечивают естественный и легкий способ разделения информации между приложениями и пользователями за счет наличия понятного человеку символьного имени и постоянства хранимой информации и расположения файла. Пользователь должен иметь удобные средства работы с файлами, включая каталоги-справочники, объеди­няющие файлы в группы, средства поиска файлов по признакам, набор команд для создания, модификации и удаления файлов. Файл может быть создан одним пользователем, а затем использоваться совсем другим пользователем, при этом создатель файла или администратор могут определить права доступа к нему других пользователей.

**6**. Классификация файловых систем?

*Ответ:*

Существуют файловые системы:

* для устройств произвольного доступа - жестких дисков;
* для устройств с последовательным доступом - магнитных лент;
* для оптических носителей - CD и DVD;
* сетевые ФС;
* виртуальные ФС.

**7.** Основная и главная философия системного программирования в Linux?

Системноепрограммирование — это практика написания системного ПО, низкоуровневый код которого взаимодействует непосредственно с ядром и основными системнымибиблиотеками. Иными словами, речь далее пойдет в основном о системных вызовах Linux и низкоуровневых функциях, в частности тех, которые определены в биб­ лиотеке C. Все доступно для редактирования.

**8**. Что такое файловый дескриптор (File Descriptor) и зачем он нужен?

*Ответ:*

Ко всем потокам ввода-вывода можно получить доступ через так называемые файловые дескрипторы. Файловый дескриптор — это неотрицательное целое число.

**9.** Обычные файлы (Regular Files) в Linux и их основные метаданные: позиция в файле (File Position), или смещение (File Offset), размер файла (File Length), имя файла (Filename), структура информационный узел (Information Node, inode)?

*Ответ:*

Это файлы, с которыми мы привыкли работать каждый день, они могут содержать текст, исполняемые инструкции для программ, изображения или другую информацию. Это самый распространенный тип файлов, которые вы можете найти в системе Linux. Рассмотрим небольшой список относящихся сюда файлов:

* Текстовые файлы
* Исполняемые файлы
* Файлы изображений
* Файлы архивов
* Файлы библиотек программ
* И другие подобные типы

**10**. Специальные файлы (Special Files) в Linux?

*Ответ:*

Специальные файлы намного интереснее, они предназначены для обмена информации с ядром, работы с устройствами или общения между программами. Такие файлы могут тоже быть нескольких типов, в зависимости от назначения.

**Блочные файлы -**это файлы устройств, которые обеспечивают буферизованный доступ к аппаратным компонентам. При записи данных на жесткий диск или на флешку нет смысла записывать данные сразу же после их поступления. Так мы будем только понапрасну расходовать ресурс устройства и энергию. Можно подождать пока наберется достаточное количество данных а потом записать их за один раз. Эти данные и собираются в буфере. С помощью таких файлов, файловая система и другие утилиты могут обращаться к драйверам аппаратных устройств. Такие файлы могут передать большой блок данных за небольшой один раз.

**11**. Что такое каталог (Directory)? Зачем нужен корневой каталог (Root Directory)?

*Ответ:*

**Каталог** (англ. **directory** справочник, указатель) — объект в файловой системе, упрощающий организацию файлов.

Root directory (корневой каталог) - основной каталог на жестком или гибком диске. Имеет фиксированный размер и местоположение для конкретного дискового тома и не может, подобно подкаталогу, динамически изменять размер. Корневой каталог, обозначаемый как “/”, является верхним элементом иерархии файловой системы. От него ответвляются все остальные каталоги, формируя “дерево файлов”. Корневой каталог (папка) - как правило - public.html корневой каталог для размешения файлов на сервере WebSphere Portal Content Publishing, применяемый для импорта и экспорта структурированной информации, а также информации на основе файлов

**12**. Что такое ссылка (Link)? Какие бывают типы ссылок?

*Ответ:*

Символические и жесткие ссылки - это особенность файловой системы Linux, которая позволяет размещать один и тот же файл в нескольких директориях.

Символические ссылки более всего похожи на обычные ярлыки. Они содержат адрес нужного файла в вашей файловой системе. Когда вы пытаетесь открыть такую ссылку, то открывается целевой файл или папка. Главное ее отличие от жестких ссылок в том, что при удалении целевого файла ссылка останется, но она будет указывать в никуда, поскольку файла на самом деле больше нет.

Этот тип ссылок реализован на более низком уровне файловой системы. Файл размещен только в определенном месте жесткого диска. Но на это место могут ссылаться несколько ссылок из файловой системы. Каждая из ссылок - это отдельный файл, но ведут они к одному участку жесткого диска. Файл можно перемещать между каталогами, и все ссылки останутся рабочими, поскольку для них неважно имя. Рассмотрим особенности:

**13**. Понятие буфера, буферизованный и расширенный файловый ввод-вывод?

*Ответ:*

Буфер - это некое промежуточное хранилище, предоставляемое для хранения временных данных либо самой программой, либо операционной системой.

Буферизированный ввод-вывод – Ввод/вывод для дисков в операционной системе MS-DOS буферизован. Это означает, что данные не сразу записываются на диск, а накапливаются в специальном массиве (буфере). По мере заполнения буфер сбрасывается на диск. При чтении информация заполняет весь входной буфер, независимо от количества байтов, которые программа читает из файла. В дальнейшем, если программе потребуются данные, которые уже были считаны с диска и записаны во входной буфер, она получит данные непосредственно из этого буфера. Обращения к диску при этом не будет.

Расширенный ввод-вывод – Расширенный ввод/вывод иногда называют дежурным вводом/выводом (alertable I/O). О том, как использовать расширенные функции, рассказывается в последующих разделах.

**14.** На какие основные части условно делиться модель файлового ввода-вывода?

*Ответ:*

При работе с файлом можно выделить следующие этапы:

* создать объект класса fstream (возможно, ofstream или ifstream);
* связать объект класса fstream с файлом, который будет использоваться для операций ввода-вывода;
* осуществить операции ввода-вывода в файл;
* закрыть файл.

**15.** Абстракция файла в языке программирования С? Стандартные файловые указатели?

*Ответ:*

Ввод или вывод информации в файлы обеспечивается с помощью так называемого указателя на файл, который является указателем на файловую структуру в памяти. При записи информации в файл или при чтении из файла программа получает необходимую информацию из структуры. Указатель на файл определяется следующим образом:

FILE \*file\_pointer;

Имя структуры FILE говорит программе о том, что определяемая переменная является указателем именно на файловую структуру. Звездочка предписывает создать указатель с соответствующим именем переменной.

Если вы собираетесь использовать одновременно несколько файлов, вам нужны указатели для каждого из них. Например, если вы пишете программу, в которой содержимое одного файла копируется в другой, вам необходимы два указателя на файлы. Два указателя требуются и в том случае, если вы хотите прочитать информацию с диска и распечатать ее на принтере:

FILE \*infile, \*outfile;

Абстра́кция в [объектно-ориентированном программировании](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — это использование только тех характеристик [объекта](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые с достаточной точностью представляют его в данной системе. Основная идея состоит в том, чтобы представить объект минимальным набором полей и методов и при этом с достаточной точностью для решаемой задачи.

**16.** Функции стандартной библиотеки С (stdio) для открытия и закрытия файлов, а также чтения и записи данных: fopen (fdopen), fread, fwrite, fclose (fcloseall), fflush, fseek, fgetc (getc or getchar), fputc (putc и putchar), fgets, gets?

*Ответ:*

Функция fopen открывает файл, имя которого указано в параметре fname и связывает его с потоком, который может быть идентифицирован для выполнения различных операций с файлом.

Функция fread() возвращает количество прочитанных элементов

А функция fwrite()возвращает количество записанных элементов.

Функция fclose закрывает и разъединяет файл filestream, связанный с потоком.

fflush – проталкивание накопленных данных в потока данных.

**fseek**– установка позиции в потоке данных

fgetc – чтение одного байта из указанного потока данных.

fputc – вывод одного байта в указанный поток данных.

fgets – чтение строки из указанного потока данных.

gets – чтение строки из стандартного потока ввода.

**17.** Форматированный ввод-вывод: printf (fprintf и sprintf), scanf (fscanf и sscanf)?

*Ответ:*

Функция printf() возвращает число выведенных символов или отрицательное значение в случае ошибки.

Функция scanf() является процедурой ввода общего назначения, считывающей данные из пото­ка stdin.

**18.** Механизм произвольного доступа к данным с использованием языка С?

*Ответ:*

В информатике под **произвольным** **доступом** понимают возможность обратиться к любому элементу последовательности за равные промежутки времени, не зависящие от размеров последовательности (в отличие от последовательного **доступа**, когда чем дальше расположен элемент, тем больше требуется времени для **доступа**).

**19.** Функция определения конца файла?

*Ответ:*

[Функция feof проверяет наличие признака конца файла](http://all-ht.ru/inf/prog/c/func/feof.html)

**20**. Функция стандартной библиотеки языка С для обработки ошибок файлового ввода-вывода?

*Ответ:*

Редко cat

Чаще stderr – exit.

**Листинг:**

* 1. **controller.cpp**

#include <stdio.h>

#include <iostream>

#include <string>

#include "model.h"

#include "view.h"

using namespace std;

int readMatrix(int rows, int cols, int\*\* a, const char\* filename)

{

FILE \*pf = fopen (filename, "r");

if (pf == NULL)

return 0;

for(int i = 0; i < rows; ++i)

for(int j = 0; j < cols; ++j)

fscanf(pf, "%d", a[i] + j);

fclose (pf);

return 1;

}

int main()

{

int m = 4;

int n = 3;

int\*\* matrix = new int\*[m];

for(int i = 0; i < m; i++)

{

matrix[i] = new int[n];

}

readMatrix(m, n, matrix, "matrix.txt");

printMatrix(matrix, m, n);

int result = getRowNumberWithLongestSeries(matrix, m, n);

printMatrixResult(result);

for(int i = 0; i < m; i++)

{

delete[] matrix[i];

}

delete[] matrix;;

return 0;

}

* 1. **model.cpp**

#include <cstdlib>

#include <iostream>

#include <string>

#include "model.h"

using namespace std;

int getLongestSeries(int\* row, int n)

{

int longestSeries = 1;

int tempSeries = 1;

for(int i = 1; i < n; i++)

{

if(row[i] == row[i - 1])

tempSeries++;

else if(tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

if(i == n - 1 && tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

}

return longestSeries;

}

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n)

{

int rowNumber = 1;

int greatestSeries = 1;

for(int i = 0; i < m; i++)

{

int series = getLongestSeries(matrix[i], n);

if(series > greatestSeries)

{

greatestSeries = series;

rowNumber = i + 1;

}

}

return rowNumber;

}

* 1. **model.h**

#pragma once

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n);

* 1. **view.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printMatrix(int\*\* matrix, int m, int n)

{

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

{

cout << matrix[i][j] << "\t";

}

cout << endl;

}

}

void printMatrixResult(int result)

{

FILE \*pf = fopen ("matrix-result.txt", "w");

if (pf == NULL)

return ;

fprintf(pf, "%s", "Row number with the longest series is ");

fprintf(pf, "%d", result);

fclose (pf);

}

* 1. **view.h**

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

* 1. **controller.java**

package by.bntu.linux.lab7;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

public class Controller {

public static void main(String[] args) {

Input input = new Input();

Printer printer = new Printer();

List<Student> students = input.readFromFile();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("What do you want to do?\n" +

"1 - show students\n" +

"2 - add student\n" +

"3 - delete student by id\n" +

"4 - sort by surname\n" +

"5 - save student to file\n" +

"6 - exit");

choice = scanner.nextInt();

switch (choice) {

case 1:

students.forEach(System.out::println);

break;

case 2:

students.add(input.inputStudent());

break;

case 3:

System.out.println("Enter id: ");

int id = scanner.nextInt();

students = students

.stream()

.filter(s -> s.getId() != id)

.collect(Collectors.toList());

break;

case 4:

students.sort(Comparator.comparing(Student::getSurname));

break;

case 5:

printer.saveToFile(students);

break;

case 6:

break;

default:

System.out.println("Invalid number!");

}

} while (choice != 6);

}

}

* 1. **student.java**

package by.bntu.linux.lab7;

import java.util.Objects;

public class Student {

private int id;

private String surname;

private String name;

private String patronymic;

private String birthDate;

private String address;

private String phone;

private String faculty;

private int course;

private int group;

public Student() {}

public Student(int id, String surname, String name, String patronymic, String birthDate, String address,

String phone, String faculty, int course, int group) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.birthDate = birthDate;

this.address = address;

this.phone = phone;

this.faculty = faculty;

this.course = course;

this.group = group;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getBirthDate() {

return birthDate;

}

public void setBirthDate(String birthDate) {

this.birthDate = birthDate;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getFaculty() {

return faculty;

}

public void setFaculty(String faculty) {

this.faculty = faculty;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id &&

course == student.course &&

group == student.group &&

Objects.equals(surname, student.surname) &&

Objects.equals(name, student.name) &&

Objects.equals(patronymic, student.patronymic) &&

Objects.equals(birthDate, student.birthDate) &&

Objects.equals(address, student.address) &&

Objects.equals(phone, student.phone) &&

Objects.equals(faculty, student.faculty);

}

@Override

public int hashCode() {

return Objects.hash(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", surname='" + surname + '\'' +

", name='" + name + '\'' +

", patronymic='" + patronymic + '\'' +

", birthDate='" + birthDate + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", faculty='" + faculty + '\'' +

", course=" + course +

", group=" + group +

'}';

}

}

* 1. **printer.java**

package by.bntu.linux.lab7;

import java.io.\*;

import java.util.List;

public class Printer {

public void saveToFile(List<Student> students) {

try(BufferedWriter writer = new BufferedWriter(new FileWriter("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

for(Student student : students) {

String line = student.getId() + " "

+ student.getSurname() + " " + student.getName() + " " + student.getPatronymic() + " "

+ student.getBirthDate() + " " + student.getAddress() + " " + student.getPhone() + " "

+ student.getFaculty() + " " + student.getCourse() + " " + student.getGroup();

writer.write(line);

}

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

}

* 1. **input.java**

package by.bntu.linux.lab7;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Input {

public List<Student> readFromFile() {

try(BufferedReader reader = new BufferedReader(new FileReader("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

String line;

List<Student> students = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(" ");

Student student =

new Student(Integer.parseInt(fields[0]), fields[1], fields[2], fields[3], fields[4], fields[5],

fields[6], fields[7], Integer.parseInt(fields[8]), Integer.parseInt(fields[9]));

students.add(student);

}

return students;

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

public Student inputStudent() {

Scanner scanner = new Scanner(System.in);

System.out.print("Surname: ");

String surname = scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Patronymic: ");

String patronymic = scanner.nextLine();

System.out.print("Birth date: ");

String birthDate = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

System.out.print("Phone: ");

String phone = scanner.nextLine();

System.out.print("Faculty: ");

String faculty = scanner.nextLine();

System.out.print("Course: ");

int course = scanner.nextInt();

System.out.print("Group: ");

System.out.print("Id: ");

int id = scanner.nextInt();

int group = scanner.nextInt();

return new Student(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

}

* 1. **view.java**

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №8

По дисциплине **«Операционные системы»**

Тема: **«IO with C++»**

**Выполнили:**

Студенты группы 10701118

Воробей

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Повторить фундаментальные основы и концепции файлового ввода-вывода в Linux, а также изучить основные функции ввода-вывода стандартной библиотеки ввода-вывода языка программирования С++ и закрепить навыки применения данной библиотеки при написании системных программ.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Предусмотреть ещё один способ инициализации данных и вывода результата работы программы – с помощью файлового ввода-вывода. Программа должна дополнительно выводить на консоль исходные данные и конечный результат.

3) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

4) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

5) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

6) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

7) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

8) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

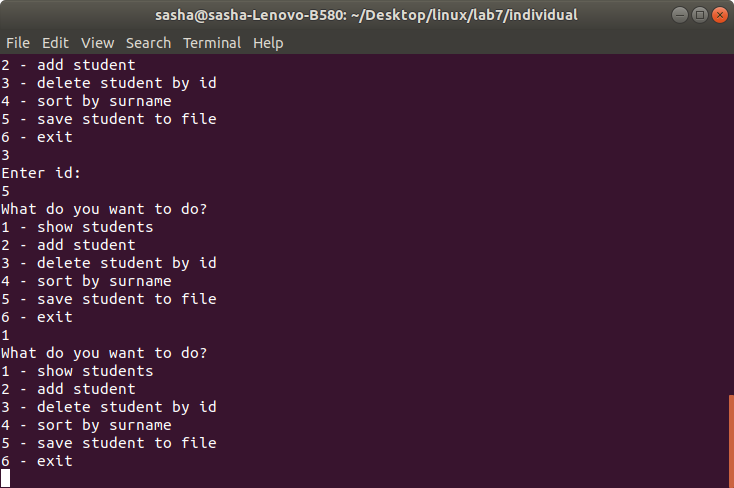
9) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

10) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Произвести рефакторинг индивидуального задания предыдущей лабораторной работы: пользовательскую библиотеку функций ввода-вывода переписать с использованием стандартной библиотеки С++.

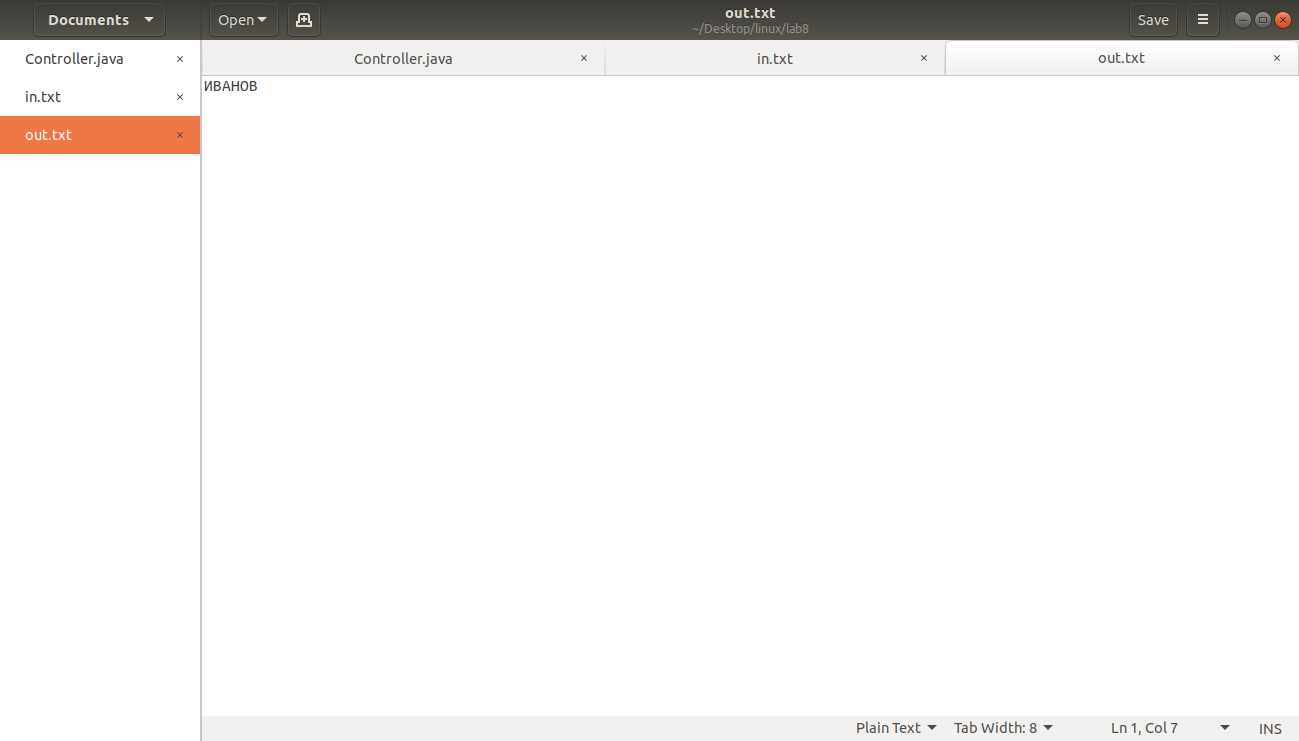
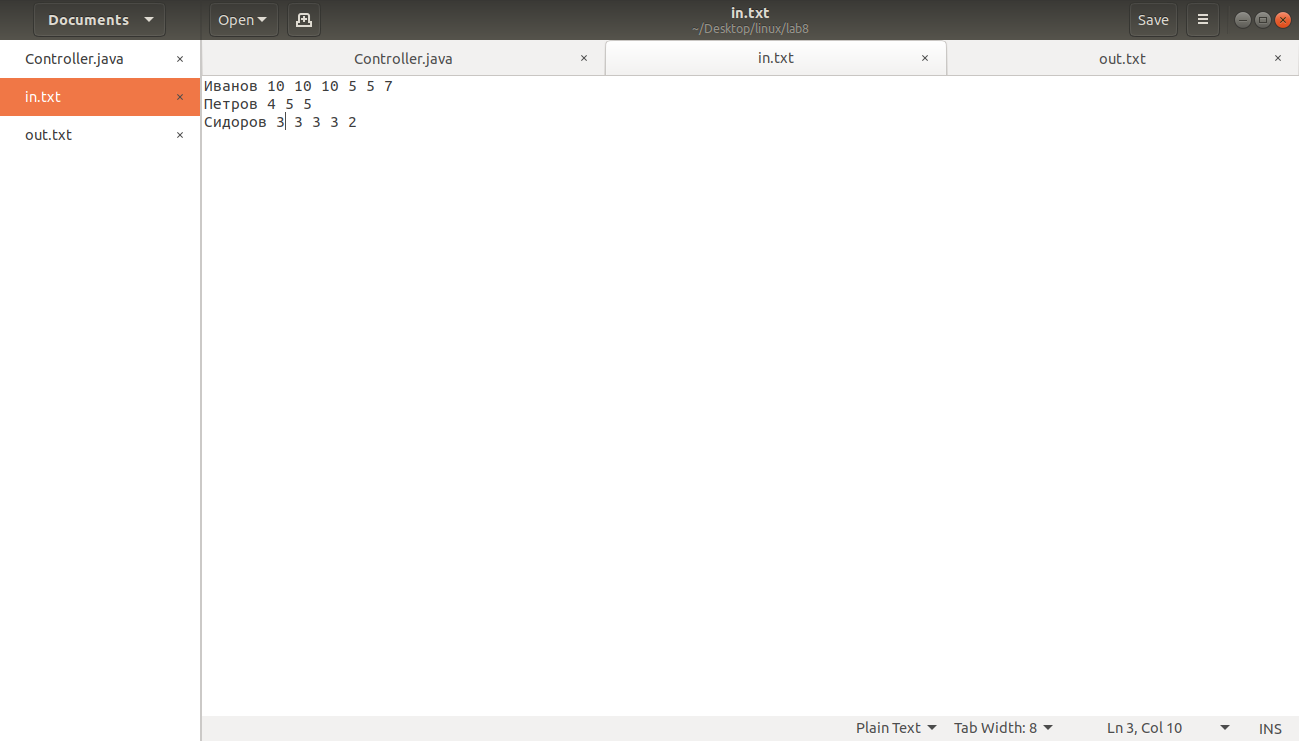
*Результаты выполнения*



**Индивидуальное задание:**

4) В файле, содержащем фамилии студентов и их оценки, записать прописными буквами фамилии тех студентов, которые имеют средний балл более “7”

*Результаты выполнения*



**Контрольные вопросы:**

**1**. Концепция файлового ввода-вывода в С++?

*Ответ:*

Есть три основных класса файлового ввода/вывода в C++:

   ifstream (является [дочерним](https://ravesli.com/urok-154-bazovoe-nasledovanie-v-c/) классу [istream](https://ravesli.com/urok-208-funktsional-klassa-istream/));

   ofstream (является дочерним классу [ostream](https://ravesli.com/urok-209-funktsional-klassov-ostream-i-ios-formatirovanie-vuvoda/));

   fstream (является дочерним классу iostream).

С помощью этих классов можно выполнять однонаправленный файловый ввод, однонаправленный файловый вывод и двунаправленный файловый ввод/вывод. Для их использования нужно всего лишь подключить [заголовочный файл](https://ravesli.com/urok-21-zagolovochnye-fajly/) fstream.

**2**. На какие основные части условно делиться модель файлового ввода-вывода в С++?

*Ответ:*

   ifstream (является [дочерним](https://ravesli.com/urok-154-bazovoe-nasledovanie-v-c/) классу [istream](https://ravesli.com/urok-208-funktsional-klassa-istream/));

   ofstream (является дочерним классу [ostream](https://ravesli.com/urok-209-funktsional-klassov-ostream-i-ios-formatirovanie-vuvoda/));

   fstream (является дочерним классу iostream).

**3**. Абстракция файла в языке программирования С++?

*Ответ:*

Абстракция данных означает предоставление только важной информации внешнему миру и скрытие их фоновых данных, т. е. Представление необходимой информации в программе без предоставления деталей.

Абстракция данных - это метод программирования (и проектирования), который основан на разделении интерфейса и реализации.

**4**. Стандартные файловые потоки?

*Ответ:*

Класс istream используется для работы с входными потоками. Оператор извлечения >> используется для извлечения значений из потока. Это имеет смысл: когда пользователь нажимает на клавишу клавиатуры, код этой клавиши помещается во входной поток. Затем программа извлекает это значение из потока и использует его.

Класс ostream используется для работы с выходными потоками. Оператор вставки << используется для помещения значений в поток. Это также имеет смысл: вы вставляете свои значения в поток, а затем потребитель данных (например, монитор) использует их.

Класс iostream может обрабатывать как ввод, так и вывод данных, что позволяет ему осуществлять двунаправленный ввод/вывод.

**5.** Функции стандартной библиотеки С++ для открытия и закрытия файлов?

*Ответ:*

.open

.close

**6**. Функции и перегруженные операторы стандартной библиотеки С++ для чтения и записи данных?

*Ответ:*

Read

Write

**7**. Форматированный ввод-вывод в С++?

*Ответ:*

Для управления вводом-выводом в C++  
используются :

* флаги форматированного ввода-вывода;
* манипуляторы форматирования.

Флаги позволяют установить параметры ввода-вывода, которые будут действовать во всех последующих операторах ввода-вывода до тех пор, пока не будут отменены. Манипуляторы вставляются в операторы **cin**(cout) и устанавливают параметры текущего оператора ввода-вывода.

**8**. Механизм произвольного доступа к данным с использованием языка С++?

*Ответ:*

В информатике под **произвольным** **доступом** понимают возможность обратиться к любому элементу последовательности за равные промежутки времени, не зависящие от размеров последовательности (в отличие от последовательного **доступа**, когда чем дальше расположен элемент, тем больше требуется времени для **доступа**).

**9.** Функция определения конца файла в С++?

*Ответ:*

eof

**10.** Функция стандартной библиотеки языка С++ для обработки ошибок файлового ввода-вывода?

*Ответ:*

Чаще stderr – exit.

**Листинг:**

* 1. **controller.java**

package by.bntu.linux.lab7;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

public class Controller {

public static void main(String[] args) {

Input input = new Input();

Printer printer = new Printer();

List<Student> students = input.readFromFile();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("What do you want to do?\n" +

"1 - show students\n" +

"2 - add student\n" +

"3 - delete student by id\n" +

"4 - sort by surname\n" +

"5 - save student to file\n" +

"6 - exit");

choice = scanner.nextInt();

switch (choice) {

case 1:

students.forEach(System.out::println);

break;

case 2:

students.add(input.inputStudent());

break;

case 3:

System.out.println("Enter id: ");

int id = scanner.nextInt();

students = students

.stream()

.filter(s -> s.getId() != id)

.collect(Collectors.toList());

break;

case 4:

students.sort(Comparator.comparing(Student::getSurname));

break;

case 5:

printer.saveToFile(students);

break;

case 6:

break;

default:

System.out.println("Invalid number!");

}

} while (choice != 6);

}

}

* 1. **student.java**

package by.bntu.linux.lab7;

import java.util.Objects;

public class Student {

private int id;

private String surname;

private String name;

private String patronymic;

private String birthDate;

private String address;

private String phone;

private String faculty;

private int course;

private int group;

public Student() {}

public Student(int id, String surname, String name, String patronymic, String birthDate, String address,

String phone, String faculty, int course, int group) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.birthDate = birthDate;

this.address = address;

this.phone = phone;

this.faculty = faculty;

this.course = course;

this.group = group;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getBirthDate() {

return birthDate;

}

public void setBirthDate(String birthDate) {

this.birthDate = birthDate;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getFaculty() {

return faculty;

}

public void setFaculty(String faculty) {

this.faculty = faculty;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id &&

course == student.course &&

group == student.group &&

Objects.equals(surname, student.surname) &&

Objects.equals(name, student.name) &&

Objects.equals(patronymic, student.patronymic) &&

Objects.equals(birthDate, student.birthDate) &&

Objects.equals(address, student.address) &&

Objects.equals(phone, student.phone) &&

Objects.equals(faculty, student.faculty);

}

@Override

public int hashCode() {

return Objects.hash(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", surname='" + surname + '\'' +

", name='" + name + '\'' +

", patronymic='" + patronymic + '\'' +

", birthDate='" + birthDate + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", faculty='" + faculty + '\'' +

", course=" + course +

", group=" + group +

'}';

}

}

* 1. **printer.java**

package by.bntu.linux.lab7;

import java.io.\*;

import java.util.List;

public class Printer {

public void saveToFile(List<Student> students) {

try(BufferedWriter writer = new BufferedWriter(new FileWriter("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

for(Student student : students) {

String line = student.getId() + " "

+ student.getSurname() + " " + student.getName() + " " + student.getPatronymic() + " "

+ student.getBirthDate() + " " + student.getAddress() + " " + student.getPhone() + " "

+ student.getFaculty() + " " + student.getCourse() + " " + student.getGroup();

writer.write(line);

}

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

}

* 1. **input.java**

package by.bntu.linux.lab7;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Input {

public List<Student> readFromFile() {

try(BufferedReader reader = new BufferedReader(new FileReader("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

String line;

List<Student> students = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(" ");

Student student =

new Student(Integer.parseInt(fields[0]), fields[1], fields[2], fields[3], fields[4], fields[5],

fields[6], fields[7], Integer.parseInt(fields[8]), Integer.parseInt(fields[9]));

students.add(student);

}

return students;

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

public Student inputStudent() {

Scanner scanner = new Scanner(System.in);

System.out.print("Surname: ");

String surname = scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Patronymic: ");

String patronymic = scanner.nextLine();

System.out.print("Birth date: ");

String birthDate = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

System.out.print("Phone: ");

String phone = scanner.nextLine();

System.out.print("Faculty: ");

String faculty = scanner.nextLine();

System.out.print("Course: ");

int course = scanner.nextInt();

System.out.print("Group: ");

System.out.print("Id: ");

int id = scanner.nextInt();

int group = scanner.nextInt();

return new Student(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

}

* 1. **view.java**

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

* 1. **controller.java**

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

public class Controller {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(new File("in.txt"));

PrintWriter pw = new PrintWriter(new File("out.txt"));

String[] student;

while (sc.hasNextLine()) {

student = sc.nextLine().split("\\s+");

double sum = 0;

for (int i = 1; i < student.length; i ++) {

sum += Integer.parseInt(student[i]);

}

if (sum / (student.length - 1) > 7) {

pw.println(student[0].toUpperCase());

}

}

sc.close();

pw.close();

}

}

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №9

По дисциплине **«Операционные системы»**

Тема: **«IO with System Call»**

**Выполнили:**

Студенты группы 10701118

Воробей

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Повторить фундаментальные основы и концепции файлового ввода-вывода в Linux, а также изучить системные вызовы (низкоуровневые функции) для взаимодействия с файловой системой и научиться использовать низкоуровневый ввод-вывод при написании системных программ.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Предусмотреть ещё один способ инициализации данных и вывода результата работы программы – с помощью файлового ввода-вывода. Программа должна дополнительно выводить на консоль исходные данные и конечный результат.

3) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

4) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

5) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

6) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

7) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

8) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

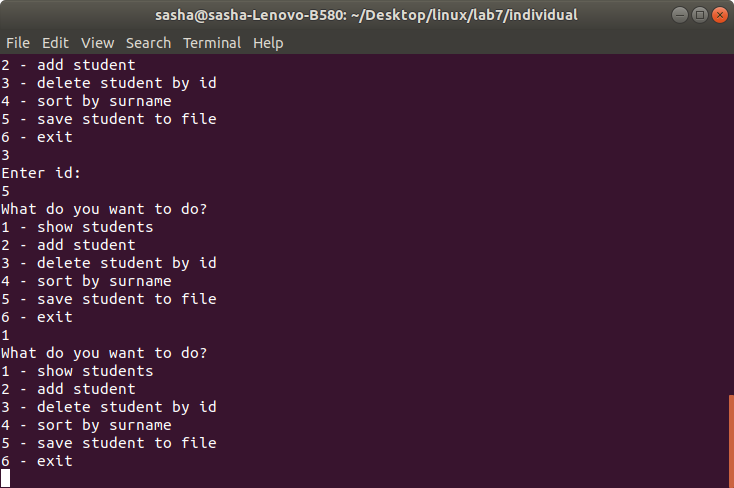
9) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

10) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Произвести рефакторинг основного задания предыдущей лабораторной работы: пользовательскую библиотеку функций ввода-вывода переписать с использованием низкоуровневых функций, которые представляет само ядро ОС.

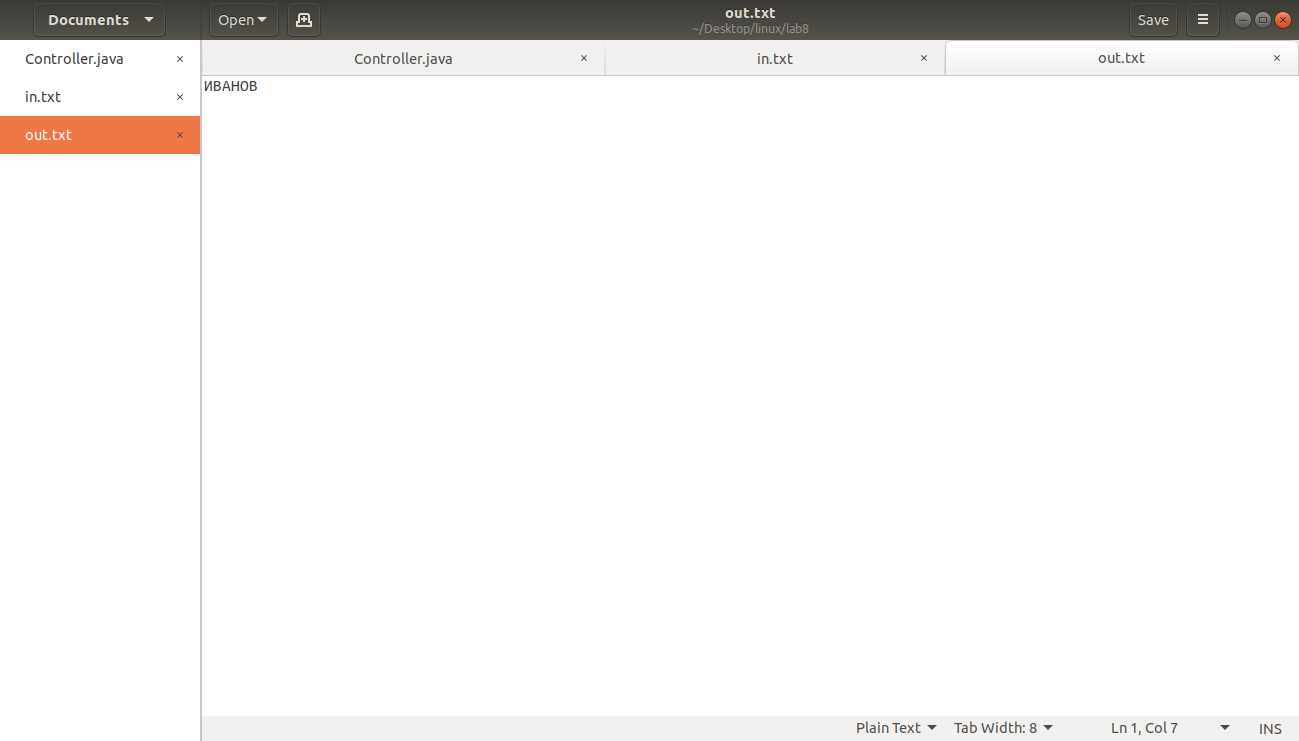
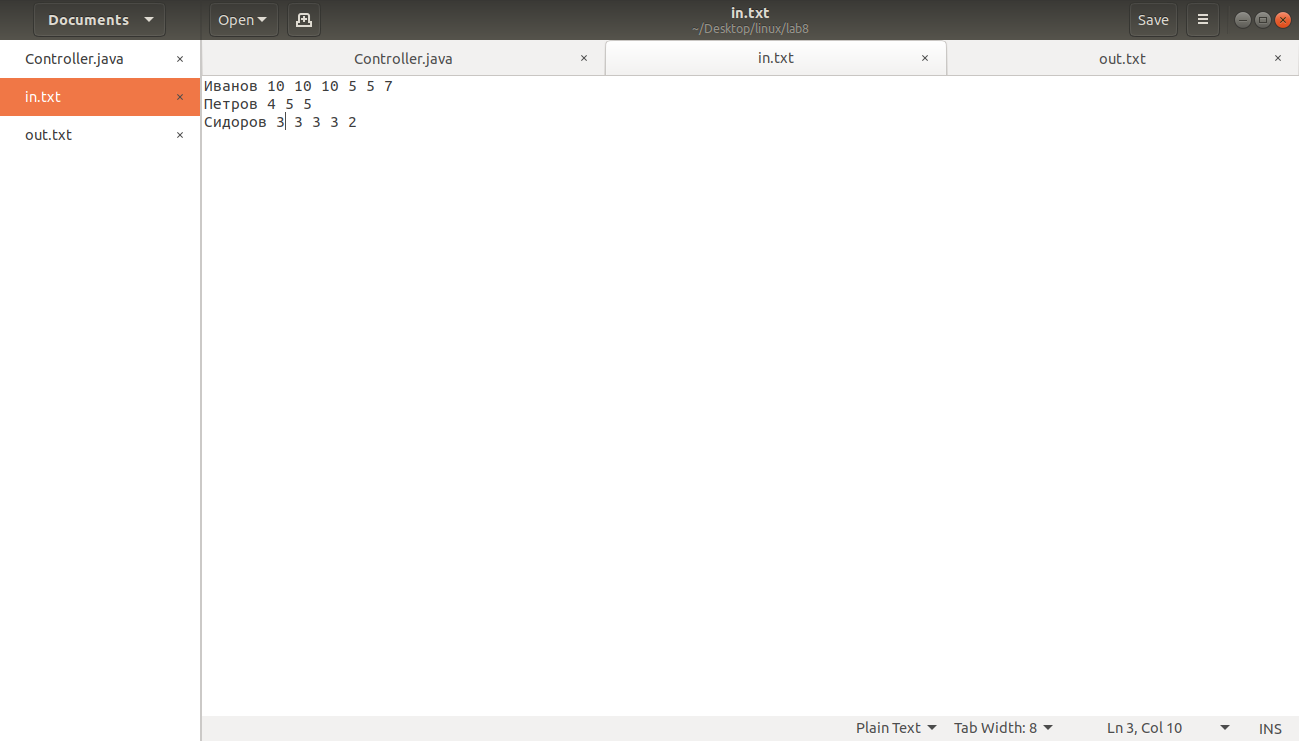
*Результаты выполнения*



**Индивидуальное задание:**

4) В файле, содержащем фамилии студентов и их оценки, записать прописными буквами фамилии тех студентов, которые имеют средний балл более “7”.

*Результаты выполнения*



**Контрольные вопросы:**

**1**. Основные концепции файлового ввода-вывода с точки зрения ОС Linux (таблица файлов, дескрипторы фалов (fd), обычные файлы, позиция в файле, смещение в файле, длина файла, усечение, информационный узел, номер inode)?

*Ответ:*

Обзор механизмов ввода-вывода в Linux. В языке C для осуществления файлового ввода-вывода используются механизмы стандартной библиотеки языка, объявленные в заголовочном файле stdio.h. Как вы вскоре узнаете консольный ввод-вывод - это не более чем частный случай файлового ввода-вывода. В C++ для ввода-вывода чаще всего используются потоковые типы данных. ... С точки зрениямодели КИС (Клиент-Интерфейс-Сервер), сервером стандартных механизмов вводавывода языка C (printf, scanf, FILE\*, fprintf, fputc и т. д.) является библиотека языка. А сервером низкоуровневого ввода-вывода в Linux, является само ядро операционной системы.

**2.** Абстракция файла и файлового ввода-вывода на уровне ядра ОС Linux (\*nix)?

*Ответ:*

Абстракция данных означает предоставление только важной информации внешнему миру и скрытие их фоновых данных, т. е. Представление необходимой информации в программе без предоставления деталей.

Абстракция данных - это метод программирования (и проектирования), который основан на разделении интерфейса и реализации.

**3**. Каталоги и ссылки в Linux (каталог, ссылка, корневой каталог, полный путь, относительный путь, записи каталога, разрешение полного пути, кэш dentry, абсолютный полный путь, рабочий каталог, жёсткие ссылки, символические ссылки, специальные файлы)?

*Ответ:*

*Каталог —* это особый файл, чье содержимое принимает форму таблицы из имен файлов в совокупности с указателями на соответствующие файлы. Эта связка из имени файла и указателя на него называется *ссылкой,* и у файлов в одном и том же или в разных ката­логах может быть несколько ссылок, а следовательно, и несколько имен.

Каталоги могут содержать ссылки как на файлы, так и на другие каталоги. С помощью ссылок между каталогами устанавливается иерархия каталогов, показанная на рис. 1.

Каждый каталог содержит как минимум две записи: . (точка), которая представляет собой ссылку на сам каталог, и .. (точка-точка), которая является ссылкой на его роди­тельский каталог — тот каталог, что расположен над ним в иерархии. Каждый каталог, за исключением корневого, имеет свой *родительский каталог.* Для корневого каталога запись .. является ссылкой на него самого (таким образом, обозначение /.. — то же самое, что и /).

Символьные ссылки

Подобно обычной ссылке, *символьная ссылка* предоставляет альтернативное имя для файла. Но, в отличие от обычной ссылки, представляющей собой в списке каталога запись вида «имя файла плюс указатель», символьная ссылка — это специально помеченный файл, содержащий имя другого файла. (Иными словами, у символьной ссылки в каталоге есть запись вида «имя файла плюс указатель», и файл, на который ссылается указатель, содержит строку с именем другого файла.) Этот последний файл часто называют *целью*символьной ссылки, и зачастую говорится, что символьная ссылка «указывает» или «ссылается» на целевой файл. Когда в системном вызове указывается путевое имя, в боль­шинстве случаев ядро автоматически снимает *косвенность* каждой символьной ссылки в путевом имени (также говорят *«следует* по ним»), заменяя ее именем того файла, на который она ведет. Этот процесс может происходить рекурсивно, если цель символьной ссылки сама по себе является символьной ссылкой. (Ядро накладывает ограничение на количество ссылок, чтобы предотвратить возможность появления замкнутых цепочек символьных ссылок.) Если символьная ссылка указывает на несуществующий файл, то говорится, что это *битая ссылка.*

В качестве альтернативных названий для обычной и символьной ссылки зачастую используются выражения *«жесткая ссылка»* и *«мягкая ссылка»*.

У каждого процесса есть свой *текущий рабочий каталог* (который иногда называют про­сто *рабочим* или *текущим).* Это «текущее местоположение» процесса в иерархии одного каталога, и именно с данного каталога для процесса интерпретируются относительные путевые имена.

Процесс наследует свой текущий рабочий каталог от родительского процесса. В случае входа в систему для оболочки рабочим каталогом является домашний каталог пользова­теля, который указан в его записи в файле паролей. Текущий рабочий каталог оболочки может быть изменен с помощью команды cd.

С каждым файлом связаны UID и GID, определяющие владельца этого файла и группу, к которой он принадлежит. Понятие «владение файлом» применяется для определения прав доступа пользователей к файлу.

**4**. Системные вызовы: ограничения системных вызовов, коды возврата системных вызовов, режим ядра, режим пользователя?

*Ответ:*

Систе́мный вы́зов (англ. system call) в программировании и вычислительной технике — обращение прикладной программы к ядруоперационной системы для выполнения какой-либо операции. Современные операционные системы (ОС) предусматривают разделение времени между выполняющимися вычислительными процессами (многозадачность) и разделение полномочий, препятствующее обращению исполняемых программ к данным других программ и оборудованию. Ядро ОС исполняется в привилегированном режиме работы процессора.

**5**. Использование системных вызовов для низкоуровневой работы с каталогами и файлами: mkdir, rmdir, chdir, opendir, closedir, readdir, readlink, link, unlink, stat, … ?

*Ответ:*

Mkdir - команда для создания новых каталогов

Rmdir - команда в операционных системах DOS, UNIX, Windows, которая удаляет каталог из файловой системы

Chdir - каманда каманднага радка, якая ўжываецца для змены бягучага каталога ў Unix, Linux, DOS, Windows, а таксама ў іншых аперацыйных сістэмах

Opendir -  открывает дескриптор каталога

Closedir – закрывает дескриптор каталога

Readdir - получить элемент каталога по его дескриптору

Readlink - считывает значение символьной ссылки

Link - команда Unix, создающая жёсткую ссылку или символьную ссылку на файл

Unlink – команда для удаления файл в системах GNU/Linux

Stat - утилита командной строки, которая отображает подробную информацию о данных файлах или файловых системах

**6**. Использование системных вызовов для низкоуровневой работы с файлом: open, creat, write, read, close, … ?

*Ответ:*

Open – открывают файл или устройство

Create –  открывают и создают файл или устройство

Write – unix-утилита, позволяющая отправить сообщение другому пользователю, который имеет сессию в этой же системе

Read – дну строку из стандартного входного потока и записывает ее содержимое в указанные переменные.

Close – закрывает файловый дескриптор, который после этого не ссылается ни на один и файл и может быть использован повторно.

**7**. Функции библиотеки С для получения описания возникших ошибок файлового ввода-вывода: perror, strerror переменная errno?

*Ответ:*

Perror – выводит в стандартный поток ошибки сообщения, описывая ошибку, произошедшую при последнем системном вызове или вызове

Strerror – функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной errno, в сообщение об ошибке, понятном человеку.

**Листинг:**

. 1) controller.java

package by.bntu.linux.lab7;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

public class Controller {

public static void main(String[] args) {

Input input = new Input();

Printer printer = new Printer();

List<Student> students = input.readFromFile();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("What do you want to do?\n" +

"1 - show students\n" +

"2 - add student\n" +

"3 - delete student by id\n" +

"4 - sort by surname\n" +

"5 - save student to file\n" +

"6 - exit");

choice = scanner.nextInt();

switch (choice) {

case 1:

students.forEach(System.out::println);

break;

case 2:

students.add(input.inputStudent());

break;

case 3:

System.out.println("Enter id: ");

int id = scanner.nextInt();

students = students

.stream()

.filter(s -> s.getId() != id)

.collect(Collectors.toList());

break;

case 4:

students.sort(Comparator.comparing(Student::getSurname));

break;

case 5:

printer.saveToFile(students);

break;

case 6:

break;

default:

System.out.println("Invalid number!");

}

} while (choice != 6);

}

}

2) student.java

package by.bntu.linux.lab7;

import java.util.Objects;

public class Student {

private int id;

private String surname;

private String name;

private String patronymic;

private String birthDate;

private String address;

private String phone;

private String faculty;

private int course;

private int group;

public Student() {}

public Student(int id, String surname, String name, String patronymic, String birthDate, String address,

String phone, String faculty, int course, int group) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.birthDate = birthDate;

this.address = address;

this.phone = phone;

this.faculty = faculty;

this.course = course;

this.group = group;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getBirthDate() {

return birthDate;

}

public void setBirthDate(String birthDate) {

this.birthDate = birthDate;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getFaculty() {

return faculty;

}

public void setFaculty(String faculty) {

this.faculty = faculty;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id &&

course == student.course &&

group == student.group &&

Objects.equals(surname, student.surname) &&

Objects.equals(name, student.name) &&

Objects.equals(patronymic, student.patronymic) &&

Objects.equals(birthDate, student.birthDate) &&

Objects.equals(address, student.address) &&

Objects.equals(phone, student.phone) &&

Objects.equals(faculty, student.faculty);

}

@Override

public int hashCode() {

return Objects.hash(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", surname='" + surname + '\'' +

", name='" + name + '\'' +

", patronymic='" + patronymic + '\'' +

", birthDate='" + birthDate + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", faculty='" + faculty + '\'' +

", course=" + course +

", group=" + group +

'}';

}

}

3) printer.java

package by.bntu.linux.lab7;

import java.io.\*;

import java.util.List;

public class Printer {

public void saveToFile(List<Student> students) {

try(BufferedWriter writer = new BufferedWriter(new FileWriter("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

for(Student student : students) {

String line = student.getId() + " "

+ student.getSurname() + " " + student.getName() + " " + student.getPatronymic() + " "

+ student.getBirthDate() + " " + student.getAddress() + " " + student.getPhone() + " "

+ student.getFaculty() + " " + student.getCourse() + " " + student.getGroup();

writer.write(line);

}

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

}

4) input.java

package by.bntu.linux.lab7;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Input {

public List<Student> readFromFile() {

try(BufferedReader reader = new BufferedReader(new FileReader("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

String line;

List<Student> students = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(" ");

Student student =

new Student(Integer.parseInt(fields[0]), fields[1], fields[2], fields[3], fields[4], fields[5],

fields[6], fields[7], Integer.parseInt(fields[8]), Integer.parseInt(fields[9]));

students.add(student);

}

return students;

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

public Student inputStudent() {

Scanner scanner = new Scanner(System.in);

System.out.print("Surname: ");

String surname = scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Patronymic: ");

String patronymic = scanner.nextLine();

System.out.print("Birth date: ");

String birthDate = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

System.out.print("Phone: ");

String phone = scanner.nextLine();

System.out.print("Faculty: ");

String faculty = scanner.nextLine();

System.out.print("Course: ");

int course = scanner.nextInt();

System.out.print("Group: ");

System.out.print("Id: ");

int id = scanner.nextInt();

int group = scanner.nextInt();

return new Student(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

}

5) view.java

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

6) controller.java

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

public class Controller {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(new File("in.txt"));

PrintWriter pw = new PrintWriter(new File("out.txt"));

String[] student;

while (sc.hasNextLine()) {

student = sc.nextLine().split("\\s+");

double sum = 0;

for (int i = 1; i < student.length; i ++) {

sum += Integer.parseInt(student[i]);

}

if (sum / (student.length - 1) > 7) {

pw.println(student[0].toUpperCase());

}

}

sc.close();

pw.close();

}}

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №10

По дисциплине **«Операционные системы»**

Тема: **«Linux Processes»**

**Выполнили:**

Студенты группы 10701118

Воробей

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Изучить фундаментальные концепции многозадачности в ОС, а также научиться порождать и управлять многозадачными приложениями в ОС Linux. **Требования:**

1) На базе полученных знаний разработать модульное консольное приложение на С/С++ согласно варианту задании A и B с использованием архитектурного шаблона проектирования MVC.

2) АПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

3) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

4) Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

5) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

6) Для компиляции, компоновки и выполнения программы использовать средства автосборки (к примеру, Makefiles и утилиту make).

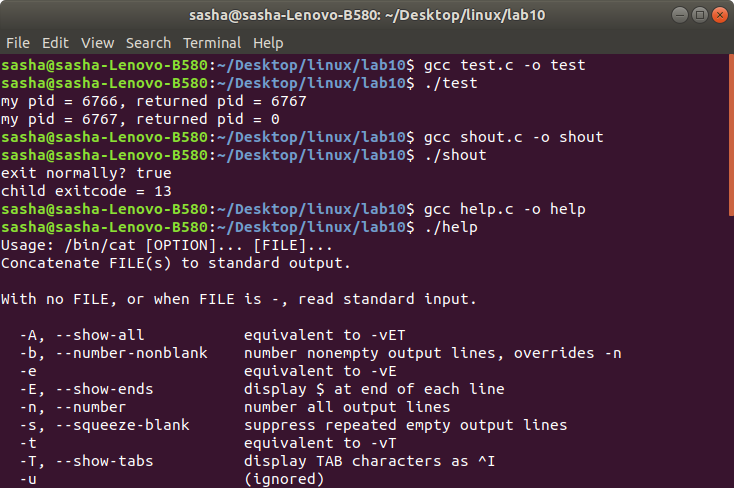
7) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

8) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Индивидуальное задание:**

1) Написать программу, которая будет превращать в демона сама себя.

*Результаты выполнения*



**Контрольные вопросы:**

**1.** Основы многозадачности (multitasking) и мультипрограммирования в системах пакетной обработки, в системах разделения времени и в системах реального времени? Мультипроцессорная обработка?

*Ответ:*

Разделяют мультипрограммирование в пакетных системах, системах реального времени и в системах разделения времени. Пакетная обработка используется для достижения максимальной эффективности использования ресурсов вычислительной машины при выполнении вычислительных задач путём сбалансированной загрузки её компонентов, как например, АЛУ и УВВ.

**2**. Понятие процесса как второй фундаментальной абстракции в Linux-системах, основные концепции, поддержка процессов ядром ОС Linux? Древовидная иерархия процессов в Linux, процесс init, Linux-команды для мониторинга процессов и просмотра их иерархии?

*Ответ:*

Процесс - это объект ОС Linux, который состоит из адресного пространства памяти и набора структур данных. По сути, процесс это запущенная программа или служба. Каждый запущенный процесс в ОС Linux может породить дополнительные процессы. Процесс, запустивший новый процесс называется родительским процессом. Новый процесс по отношению к создавшему его процессу называется дочерним. Процессы - это не то же самое, что задачи: процессы являются частью операционной системы.

**3**. Планирование процессов и их приоритет? Уступчивость процесса и использование системного вызова nice?

*Ответ:*

Знание алгоритмов **планирования** **процессов** **и** системы **приоритетов** ОС Unix во многих случаях позволяют **системному** администратору успешно справиться с проблемами повышения производительности. В случае если компьютер, работающий под управлением ОС Unix, решает задачи в реальном времени, значение **планирования** **процессов** еще больше возрастает. В предлагаемой статье дан обзор вопросов, связанных с **приоритетами** **процессов** **и** **их** **планированием** в ОС Unix. Однако ряд моментов **планирования**, специфических для мультипроцессорных систем, в статье не рассматривается.

**4.** Структура данных Linux-процесса, идентификатор и атрибуты процесса? Способы получения информации о процессе? Использование соответствующих системных вызовов? Как получить имя пользователя из его числового имени, т.е. из UID?

*Ответ:*

Атрибуты процесса. Процесс в ядре представляется просто как структура с множеством полей (определение структуры можно прочитать здесь). Но так как статья посвящена системному программированию, а не разработке ядра, то несколько абстрагируемся и просто акцентируем внимание на важных для нас полях процесса: Идентификатор процесса (pid). Открытые файловые дескрипторы (fd). Обработчики сигналов (signal handler).

**5**. Какие бывают типы процессов (порождающий (родитель), порождённый (потомок), демон, зомби, приведение)?

*Ответ:*

Демоны и потоки. Среди всех процессов можно выделить несколько особых типов процессов. Системные процессы являются частью ядра и всегда находятся в оперативной памяти. ... Демоны отличаются от обычных процессов только тем, что они работают в неинтерактивном режиме. Если с обычным процессом всегда ассоциирован какой-то терминал или псевдотерминал, через который осуществляется взаимодействие процесса с пользователем, то демон такого терминала не имеет. ... Но бывают случаи, когда родительский процесс завершается раньше дочернего. Процессы, не имеющие родителя, называются "сиротами".

**6**. Состояние процесса, время выполнения процесса, понятие очереди обработки процессов? Подходы, используемые для обработки очередей: квантование, приоритет, смешанный?

*Ответ:*

Например, приоритет очереди системных процессов устанавливается выше, чем приоритет очередей пользовательских процессов. А приоритет очереди процессов, запущенных студентами, ниже, чем для очереди процессов, запущенных преподавателями.

**7**. Как осуществить порождение дочернего процесса? Концепция развилки, создание (ветвление) и исполнение новых процессов (передача управления)? Использование системных вызовов fork (vfork) и семейства вызовов exec (execl, execlp, execle, execv, execvp, execve)?

*Ответ:*

Так как любой процесс содержит хотя бы одну нить, создание (т.е. порождение) и завершение процессов подразумевает создание и завершение нитей. ... Для создания нового потока управления в рамках уже существующей программы. AIX создает новый процесс. Для создания нового процесса, выполняющего другую программу. В этом случае вскоре после вызова функции fork вызывается одна или несколько функций exec. В программе с несколькими нитями новые потоки управления создаются не функцией fork, а функцией pthread\_create. Функцию fork следует применять только для запуска программ.

**8**. Как осуществить ожидание выполнения дочернего процесса? Использование системных вызовов wait и waitpid?

*Ответ:*

wait, waitpid - ожидает завершения процесса. СИНТАКСИС. #include <sys/types.h> #include <sys/wait.h>. ... Функция wait приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится, или до появления сигнала, который либо завершает текущий процесс, либо требует вызвать функцию-обработчик. Если дочерний процесс к моменту вызова функции уже завершился (так называемый "зомби" ("zombie")), то функция немедленно возвращается. Системные ресурсы, связанные с дочерним процессом, освобождаются.

**9**. Как завершить выполнение процесса? Использование системного вызова \_exit и библиотечной С-функции exit? Что произойдет с работающим потомком при завершении родительского процесса?

*Ответ:*

Системный вызов \_exit завершает процесс, он вызывает освобождение сегмента данных процесса, сегмента стека, распределения областей и закрытия всех открытых дескрипторов файловю. Но запись в таблице процессов, в которой был зарегистрирован этот процесс, не удаляется, чтобы там был зарегистрирован статус завершения процесса и информация о его выполнении. Такой остаток процесса называется процессом - “зомби”. ... Системные функции типа exec. Существует группа системных функций, которые предназначаются для перезапуска программы. #include<unistd.h>. ... exit (0); } } Системный вызов exec() заставляет вызывающий процесс изменить свой контекст и выполнить другую программу.

**10**. В каких целях используют библиотечную функцию system? Как она работает?

*Ответ:*

Электронно-библиотечная система (ЭБС) — это предусмотренный федеральными государственными образовательными стандартами высшего профессионального образования (ФГОС ВПО) России обязательный элемент библиотечно-информационного обеспечения учащихся вузов, представляющий собой базу данных, содержащую издания учебной, учебно-методической и иной литературы, используемой в образовательном процессе.

**Листинг:**

* 1. **help.cpp**

#define \_GNU\_SOURCE

#include <unistd.h>

int main() {

char\* args[] = { "/bin/cat", "--help", NULL };

execve("/bin/cat", args, environ);

return 1;

}

* 1. **shout.cpp**

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

int pid = fork();

switch(pid) {

case -1:

perror("fork");

return -1;

case 0:

// Child

return 13;

default: {

// Parent

int status;

waitpid(pid, &status, 0);

printf("exit normally? %s\n", (WIFEXITED(status) ? "true" : "false"));

printf("child exitcode = %i\n", WEXITSTATUS(status));

break;

}

}

return 0;

}

* 1. **test.cpp**

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

int pid = fork();

switch(pid) {

case -1:

perror("fork");

return -1;

case 0:

// Child

printf("my pid = %i, returned pid = %i\n", getpid(), pid);

break;

default:

// Parent

printf("my pid = %i, returned pid = %i\n", getpid(), pid);

break;

}

return 0;

}